UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA ADMINISTRACION

ESCUELA DE INGENIERIA DE SISTEMAS

TECNOLOGIA SOA "ARQUITECTURA ORIENTADA A SERVICIOS"

MONOGRAFIA PREVIA A LA OBTENCION DEL TITULO DE INGENIERO DE SISTEMAS

AUTOR: CARLOS XAVIER GONZALEZ MEJIA

DIRECTOR: ING. PABLO PINTADO

CUENCA – ECUADOR

2007

DEDICATORIA

A mi esposa y mis padres con mucho cariño.

AGRADECIMIENTO

Principalmente a mis padres, por su compresión, apoyo y sacrifico para lograr culminar mi formación profesional. A todos los miembros de mi familia y amigos que siempre estuvieron para apoyarme, un agradecimiento sincero. A mi director de monografía el Ing. Pablo Pintado por su paciencia y acertada dirección, a la Universidad del Azuay, a su cuerpo docente y a todos quienes directa e indirectamente me han apoyado en este trabajo.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

RESUMEN

Esta monografía tiene por objetivo principal, el acercarnos al conocimiento de SOA (Arquitectura Orientada a Servicios), que consiste en una plataforma para el diseño y desarrollo de sistemas que proveen servicios a otras aplicaciones, a través de interfaces identificables y publicables, generando servicios que pueden ser invocados desde la red. SOA permite que las organizaciones combinen fluidamente sus activos de TI existentes para desarrollar nuevas aplicaciones, procesos y modelos de negocios internos o externos a la empresa. Es definida SOA, sus antecedentes, el porque de su auge, desventajas, su arquitectura, Servicios Web, condiciones para su implementación y posibles dificultades.

ABSTRACT

The main objective of this research paper is to study the SOA (Services Orientated Architecture) which consists in a platform for the design and development of systems that provide services to other applications through identifiable and publishable interfaces, generating services that can be invoked from the net. SOA allows organizations to combine smoothly their existing TI assets in order to develop new business applications, processes, and models either inside or outside the company. The study includes a definition of SOA as well as a discussion of its background, the reason of its popularity, its disadvantages, architecture, Web services, conditions for its implementation, and possible difficulties.

INTRODUCCIÓN

Arquitecturas Orientadas a Servicio (Service-Oriented Architectures -SOA) es un término que se menciona con mucha frecuencia en gran parte de las publicaciones relacionadas con IT y además por las empresas proveedoras de plataformas de aplicaciones.

Muchos programadores y arquitectos han sido inducidos a diseñar y programar de acuerdo con los lineamientos que indica este nuevo paradigma, entusiasmados ante la posibilidad de resolver, de una vez y para siempre, algunos de los problemas que se tiene desde hace años: la duplicación de soluciones, la interoperabilidad y la interconexión de aplicaciones.

El presente trabajo investigativo tiene como objetivo hacer un recorrido general e imparcial por lo que existe detrás de esta Arquitectura Orientada a Servicio. La metodología a utilizarse en este trabajo es mediante la consulta bibliográfica y sitios de Internet especializados en este tema.

Nos centraremos en la definición conceptual de SOA, su antecedentes, su presente, su futuro, el porqué de su auge, las ventajas y desventajas que conlleva su uso, las condiciones bajo las cuales podemos obtener beneficios reales en su implementación, los métodos y los requerimientos para su adopción, y las dificultades que podemos encontrar en el proceso.

CAPITULO 1

1. ¿QUÉ ES SOA?

En su libro Service-Oriented Architecture el autor Thomas Erl define la arquitectura orientada a servicios de la siguiente manera:

"SOA contemporánea representa una arquitectura abierta, ágil, extensible, federada, armable, compuesta de servicios que son autónomos, con capacidad de dar calidad de servicio, de diversos vendedores, ínter operables, descubribles y potencialmente reusables, implementados con Web Services.
SOA puede establecer una abstracción de lógica de negocios y tecnología que permita introducir cambios en el modelado de procesos de negocios y arquitectura tecnológica, resultando en un bajo acoplamiento entre estos modelos".

SOA es una Arquitectura o Plataforma y a la vez, un Modelo - Filosofía para el diseño y desarrollo de sistemas que proveen servicios a otras aplicaciones a través de interfaces identificables y publicables, generando servicios que pueden ser invocados desde la red.

En la implementación de una plataforma SOA que usa Tecnología Web Services, se generan nuevas formas de construir aplicaciones, con modelos flexibles.

Es el próximo paso hacia lo que se considera como solución de manera que se pueda dar respuesta a las necesidades de los negocios de adaptar soluciones tecnológicas flexibles y poder asumir los nuevos retos.

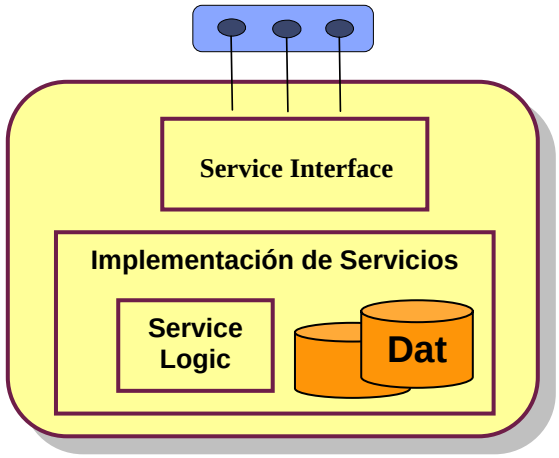
A lo largo de este trabajo investigativo analizaremos todo lo referente a esta arquitectura de la cual mucho se habla, su relación con los Web Services, cuáles son las plataformas que la soportan, la manera de implementarla y qué consideraciones se deben tener en el uso e implementación de las herramientas que demanda esta arquitectura.

Servicio es uno de los términos mas utilizados en SOA; por lo tanto, es conveniente aclarar el significado que se le da, en este contexto, a esta palabra, que puede tener varias interpretaciones.

1.1 Definición de servicio

Desde el punto de vista de las arquitecturas de software, el término servicio ha sido utilizado, tradicionalmente, para describir una función de negocio autocontenida, con una interfaz bien definida y estable, que recibe requerimientos de sus clientes. El servicio no depende del contexto de sus clientes y puede ser consumido por varios sistemas sin ser modificado. Los servicios son instalados (o desplegados) una única vez (esto los diferencia de los componentes que deben ser incluidos dentro del contexto de cada aplicación que requiera de su uso) y permanecen disponibles, sin consumir recursos, hasta que son invocados.

Figura 1. Estructura de un Servicio



Fuente: (4)

Desde el punto de vista tecnológico, SOA propone varias capas de servicios que exponen funcionalidad, fundamentalmente de negocio, que permiten la composición de aplicaciones a partir de los mismos.

“En el nivel más bajo, tendremos una capa de servicios de “Acceso a Información y Datos”, cuya función es exponer en un ESB (Enterprise Service Bus, por sus siglas en inglés) la funcionalidad y facilidades provistas por los sistemas corporativos. Muchas veces, estos servicios básicos no representan realmente “servicios de negocio” que aporten real valor al resto de la organización, al menos no por sí solos. Por este motivo que sobre esta primera capa, encontraremos una segunda de servicios que denominamos de “Servicios de Negocio Compartidos” que

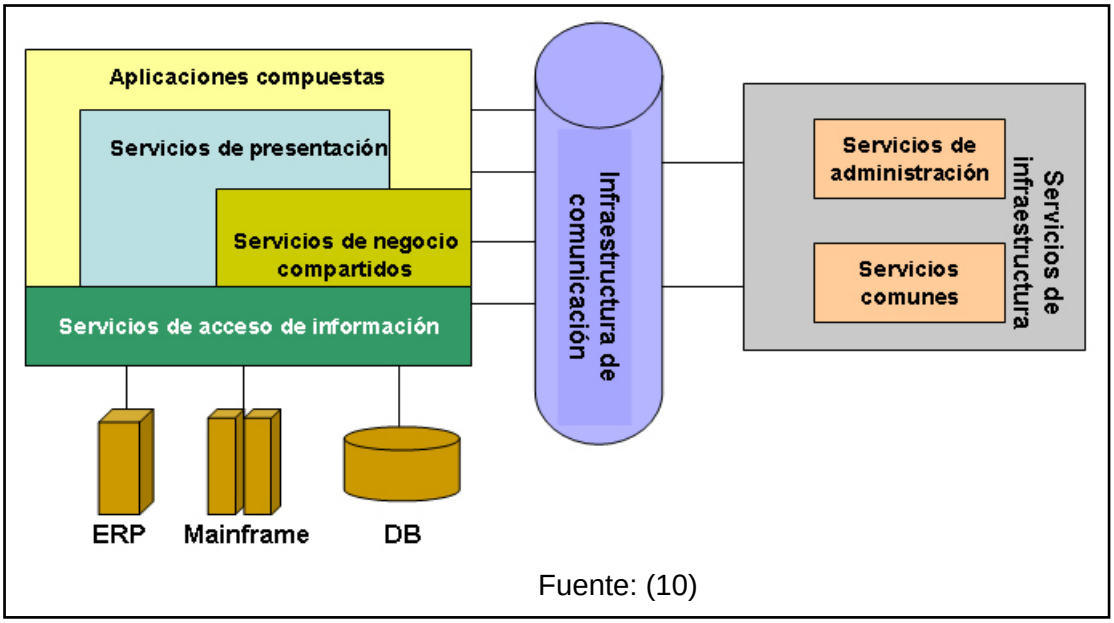
componen y enriquecen la funcionalidad expuesta en la capa precedente, con el objeto de agregar valor desde el punto de vista del negocio de una organización."(9)

Con base en estos servicios básicos y de negocio, es posible comenzar el desarrollo de aplicaciones compuestas. Estos desarrollos son los que aprovechan los beneficios de un esquema SOA en cuanto a reusabilidad de servicios existentes y adaptabilidad al negocio. Sin embargo, es necesario que en esta capa, las aplicaciones también expongan en el Bus de Servicios la funcionalidad que resuelven, para que sean aprovechadas por otras aplicaciones y servicios.

En arquitecturas que avanzan en este paradigma lo suficiente, encontraremos servicios que se exponen incluyendo su propia lógica de presentación. El caso más representativo es el de "Portlets"; aplicaciones -o más bien componentes funcionales-cuya presentación se visualiza en una ventana que puede ser incluida y reutilizada en distintos Portales. Este tipo de servicios se publican en la capa de "Servicios de Presentación".

Finalmente, y para dar soporte a la arquitectura en su conjunto, encontraremos una serie de servicios de infraestructura, que proveen facilidades de seguridad, administración y soporte al esquema completo, y que es fundamental cuando la arquitectura aumenta en volumen y complejidad.

Figura 2. Capas de servicios



Fuente: (10)

“Imaginemos, en este punto, un esquema que se reitera usualmente en aplicaciones de Banca, en las que un conjunto de transacciones CICS Cobol exponen la funcionalidad del core bancario en el mainframe. ¿Cuál es la diferencia entre esa arquitectura y lo que hoy nos propone SOA? Esto tiene fundamentalmente que ver con el estado actual de la tecnología y los estándares. Acceder a los servicios CICS Cobol e integrarlos a distintos lenguajes y tecnologías no era una tarea trivial. Exponer estos servicios a terceras empresas involucraba acuerdos complejos e integraciones punto a punto difícil de mantener y administrar. Una de las diferencias claves en la adopción de SOA es la utilización de tecnologías basadas en estándares que faciliten e incluso permitan automatizar la publicación e integración de aplicaciones y servicios. “(9)

Web Services no es se considera un sinónimo de SOA, ya que es posible utilizar Web Services y seguir en un esquema de integración punto a punto, así como es posible implementar un esquema SOA sin utilizar Web Services.
No obstante, tanto Web Services como los estándares asociados, XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), UDDI (Universal Description, Discovery, and Integration) conforman una base de tecnologías y estándares que facilitan la implementación de SOA.

1.2 Propiedades de un servicio

- Los servicios deben estar disponibles a través de protocolos ínter operables, de forma que puedan ser consumidos por cualquier sistema. Este principio se suele lograr mediante el uso de lenguajes y protocolos estándar que proporcionan independencia de plataforma.

- Los servicios deben estar accesibles y deben poder ser localizados para su consumo. Esto se suele resolver mediante el uso de repositorios de servicios que permitan tener una visión fiable de qué servicios hay disponibles y cómo pueden ser accedidos.

- Los servicios deben ser reutilizables y reutilizados, es decir, se debe buscar la reutilización de servicios ya disponibles. La disponibilidad de un registro y de mecanismos de búsqueda de servicios, así como el uso de estándares que faciliten

el uso de servicios existentes, aumentan las posibilidades de una reutilización real de los activos tecnológicos.

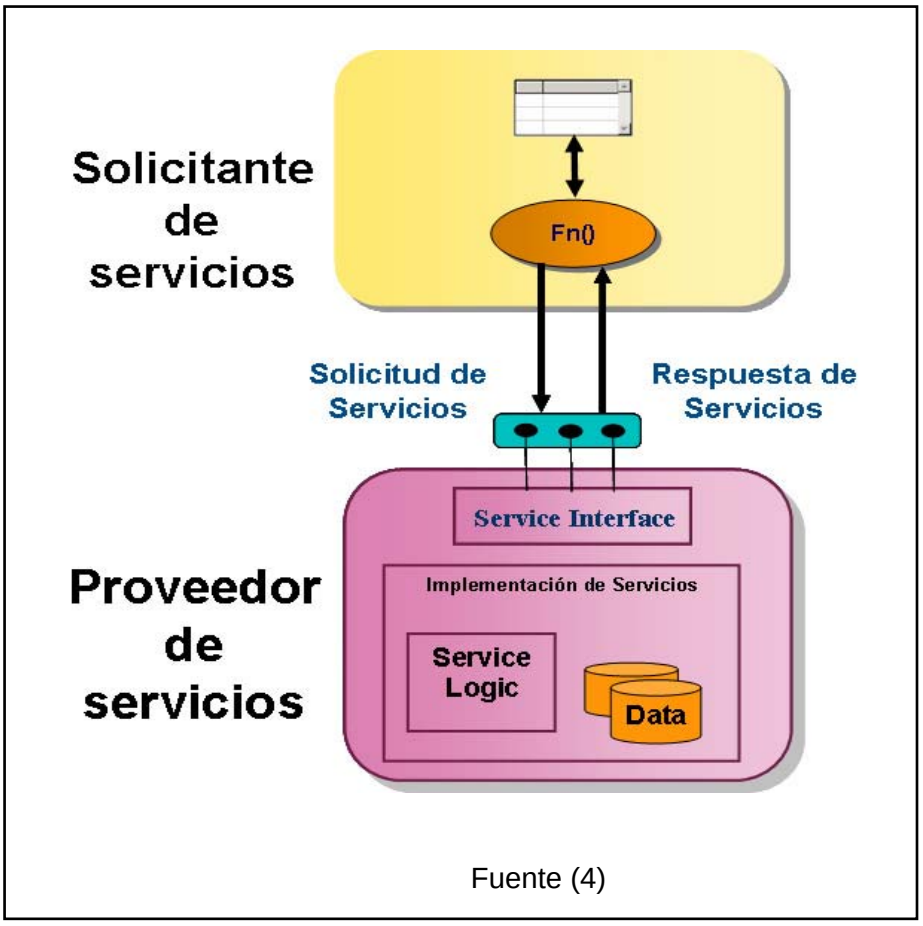
- Los servicios no exponen detalles de implementación, sino que deben proporcionar una interfaz estable y basada en estándares independientemente de su implementación, de forma que cambios en la implementación no afecten a los consumidores del servicio.

- La funcionalidad expuesta por un servicio debe ser relevante, es decir, debe tener la granularidad adecuada.

- Además debemos considerar otras propiedades de lo servicios:

- Interfaz bien definida
- Autocontenido
- No depende del contexto de sus clientes
- No requiere ser desplegado con cada cliente

Figura 3. Flujo de servicios



Fuente (4)

1.2.1 Del servicio a la arquitectura

Consideramos un estilo arquitectónico que propone modelar la empresa como una colección de servicios expuestos en la red.

Del mismo modo que en algún momento el paradigma de orientación a objetos nos hizo cambiar la manera de pensar en el análisis, el diseño y la construcción de aplicaciones, SOA se propone cambiar la manera en que se concibe la arquitectura, no ya de una aplicación o de un sistema aislado, sino de la empresa como un todo.

Sobre la base de que toda funcionalidad de negocio está expuesta y es accesible a través de la red bajo la forma de un servicio, las nuevas aplicaciones no serán desarrolladas “desde cero”, sino que serán un “ensamble” particular de un conjunto de servicios publicados por múltiples proveedores, internos o externos a la organización.

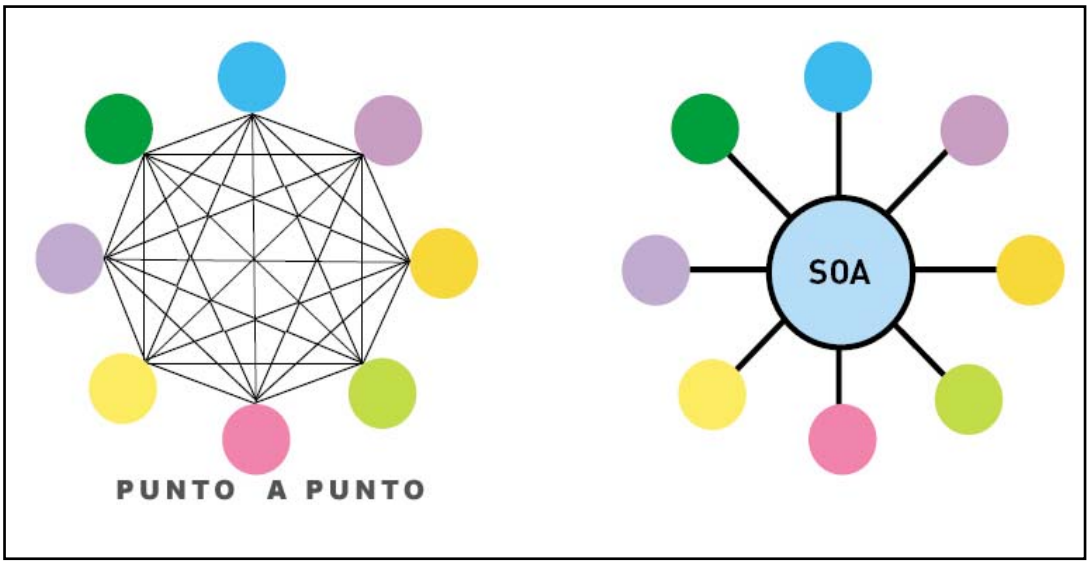
En la mayoría de las empresas, la proliferación de aplicaciones lleva al surgimiento espontáneo de una arquitectura corporativa “accidental”, esto es, no planeada. En estos entornos es usual la existencia de varios proyectos de integración, que intentan solucionar los problemas surgidos de la interacción necesaria entre aplicaciones.

SOA intenta ser un modelo sobre el cual las organizaciones puedan construir una arquitectura corporativa de referencia, que contemple y favorezca la interacción de aplicaciones.

Para lograr este objetivo, los servicios componentes de la arquitectura deberán ser implementados basándose en protocolos estándar que garanticen interfaces independientes de la implementación, y transparencia en la ubicación de los servicios.

Como podemos observar en la siguiente figura la integración tradicional Punto a Punto casi siempre redunda en costos muy altos, además de ser poco escalable. SOA nos da la alternativa de un modelo en que la integración se realiza por medio de un canal común, basado en estándares que permitan una mayor interoperabilidad

Figura 4. Punto a Punto vs. SOA



1.3 Distintas cosas para distinta gente

Es posible tener una percepción distinta del mismo concepto, dependiendo del punto de vista que se emplee.

Un ejecutivo del negocio puede ver a SOA como un conjunto de servicios de negocio que su organización busca exponer a sus clientes, socios o a otras áreas que la integran. Para un arquitecto de software, en cambio, SOA será un estilo Arquitectónico que va más allá de la implementación, al definir principios de diseño, patrones y criterios que permiten lograr características tales como modularidad, encapsulamiento, bajo acoplamiento, separación de responsabilidades, reuso, composición, etc.

Finalmente, para un desarrollador, SOA es un modelo de programación que provee estándares, herramientas y tecnologías concretas, que le permiten llevar a cabo su tarea diaria.

CAPITULO 2
¿POR QUÉ SOA?

Analizaremos y trataremos de dar respuesta a esta pregunta, en base a lo que se considera como motivaciones y necesidades del negocio, y las ventajas de utilizar SOA en una arquitectura empresarial.

2.1 Motivaciones y necesidades del negocio

Desde hace ya algún tiempo que los sistemas y las redes de computadoras dejaron de ser solamente herramientas para dar soporte a los negocios que se ejecutan en el mundo físico. En la actualidad, la red es un medio de negocios en sí misma.

La interoperabilidad prometida por SOA permite “negociar” en esta red de un modo estándar y, así, reducir los costos de integración.

La manera en que una empresa produce y comercializa sus bienes y servicios, y el modo en el que se interrelaciona con proveedores, socios, clientes y empleados, definen y constituyen su negocio. En otras palabras, los procesos de una empresa son su negocio.

Tomando en cuenta que los negocios pueden cambiar a una velocidad asombrosa, los sistemas de TI, que deberían ser facilitadores del cambio, casi siempre se convierten en un verdadero impedimento. Implementar cambios en un proceso de negocio puede demorar meses y además generar enormes costos.

El modelo de servicios propuesto por SOA permite a las empresas replicar sobre la red los procesos con los que ejecutan sus negocios en el mundo real, generando aplicaciones y sistemas mucho más cercanos a sus necesidades y a las de los usuarios. SOA se propone cerrar la brecha existente entre los procesos de negocios y las implementaciones de TI, y permitir la flexibilidad, la eficiencia y la velocidad de adaptación necesarias para soportar los continuos cambios del mundo de los negocios.

Sólo el transcurrir del tiempo dirá si SOA es capaz de lograr cumplir con las expectativas fijadas. Mientras tanto, está logrando la adhesión por parte de los

principales proveedores de plataformas de aplicaciones, empresas, arquitectos y programadores de la comunidad de sistemas, como para transformarse, en breve, en una exigencia para cualquier profesional de TI.

2.2 Ventajas

En todo proceso de arquitectura empresarial robusto se tiene que responder preguntas como:

¿La arquitectura actual está dando soporte y agregando valor a la organización? ¿Cómo debería ser modificada la arquitectura para que agregue más valor aún? ¿Podrá la arquitectura actual soportar los objetivos que persigue la organización para el futuro?

La adopción gradual de SOA dentro de una organización se presenta como una respuesta posible a estos interrogantes. Para lograrlo, se intenta crear un concepto, una tecnología y un marco de procesos que permita a las empresas desarrollar, interconectar y mantener aplicaciones empresariales y servicios de manera eficiente y económica.

Considerar los sistemas como una composición de servicios independientes es, en sí mismo, todo un desafío que promueve un cambio de mentalidad en todos los niveles de la organización. ¿Por qué hacerlo? ¿Qué beneficios podemos obtener si decidimos implementarlo?

Los promotores de SOA enumeran una serie de ventajas que justifican la adopción de este enfoque. Si efectivamente, SOA promueve y facilita ciertas características deseables en todo sistema, no es menos cierto que muchos de estos beneficios no son exclusivos de esta arquitectura, sino que también pueden obtenerse de la manera tradicional; con un buen diseño.

Las siguientes hemos considerado como ventajas y las enunciamos para su consideración:

- Reduce el nivel de acoplamiento:

La definición de interfaces minimiza el acoplamiento, con lo cual permite que las modificaciones en la implementación de un servicio no afecten a sus consumidores.

- Mejora la definición de los roles de desarrollo:

La implementación de cada función del negocio como un servicio independiente permite delimitar claramente las responsabilidades de cada desarrollador.

- Permite una delineación de seguridad más clara:

La definición de seguridad puede ser realizada a nivel de servicios, con lo que se logra una granularidad más fina al otorgar permisos que cuando éstos se administran a nivel aplicación.

- Facilita el testeo:

La utilización de la modularización y el bajo acoplamiento permiten la definición de pruebas integrales mucho más precisas, y de esta manera se logra mejorar la calidad del código producido.

- Mejora la mantenibilidad:

Ubicar toda la lógica de negocio en la capa de servicios simplifica la “arqueología de software”; es decir, la dura tarea de localizar errores.

- Favorece el reuso y mejora la productividad:

La implementación de servicios que permitan la interoperabilidad –algo difícil de conseguir con otras técnicas de reuso y la publicación de sus interfaces permiten un alto nivel de reuso, al ofrecer una forma concreta de reutilizar funcionalidades completas del negocio.

- Favorece el desarrollo en paralelo:

Dado que cada servicio es una unidad funcional independiente, una vez definidas sus interfaces, es posible atacar simultáneamente la implementación de varios servicios.

- Mejora la escalabilidad y la alta disponibilidad:

La transparencia en la localización de los servicios (implementada mediante un directorio en el que cada proveedor publica sus servicios) permite que, ante una demanda excesiva o una falla en un servidor o en un segmento de la red, los requerimientos sean redireccionados hacia otros equipos de un modo transparente para la aplicación cliente.

- Logra un mapeo más directo entre los procesos de negocio y los sistemas:

La composición de servicios permite replicar los procesos del negocio del mundo físico en el mundo virtual, de un modo más directo y natural.

- Permite un monitoreo más preciso de los procesos:

Dado que los servicios tienen puntos de acceso bien definidos, resulta más sencillo monitorear de un modo preciso los procesos del negocio, con el fin de detectar inconvenientes de manera temprana y poder tomar las medidas correctivas del caso.

- Permite la interoperabilidad:

En la medida en que nuestra capa de servicios se encuentre basada en protocolos estándar (un requisito fundamental de una arquitectura SOA), podremos obtener la posibilidad de crear aplicaciones compuestas de múltiples servicios que operen entre sí, independientemente del lenguaje que se haya utilizado para la implementación o de la plataforma en donde éstos se ejecuten.

Aparte de las ventajas técnicas, existen una serie de motivaciones y necesidades insatisfechas, desde el punto de vista de las organizaciones, que conducen (inevitable y periódicamente) a un replanteo en la concepción tradicional de los sistemas con el fin de satisfacerlas.

CAPITULO 3
MÁS ALLÁ DEL AUGE DE SOA

3.1 Antecedentes

Desde sus inicios, la Organización OMG (Object Management Group), generó la posibilidad de trabajar con programas que eran independientes de sus plataformas, proveedores, etc. convirtiéndose en los primeros Middleware, llamados en ese entonces ORB´s (Object Request broker) los cuáles se basaron en especificaciones CORBA (Common object request broker architecture).

Lamentablemente no todos los sistemas se desarrollaron usando esta filosofía y se incrementaron los problemas de integración.

La aparición de Java contribuyó a plataformas de programación neutral y XML contribuyó en datos que se autodescriben en plataformas neutrales.

La aparición de Web Services eliminó algunas barreras, permitiendo la interconexión de aplicaciones mediante modelos orientado a objetos.

Esto permite que a cada servicio requerido se obtenga una respuesta:

- Sin importar donde están las aplicaciones.
- En que lenguaje fueron desarrolladas.
- La ruta que el mensaje debe seguir para ser exitoso

SOA no es un concepto nuevo, desde hace más de 30 años la industria viene planteándose la necesidad de crear arquitectura de software débilmente acopladas. CORBA es muy probablemente la más cercana encarnación de este concepto.

Desde hace ya algún tiempo que las empresas proveedoras de software, los organismos de estandarización y mucha gente alrededor del mundo intentan resolver estos problemas y aportar los beneficios enumerados previamente.

3.2 Otras alternativas

No todo es nuevo por completo. Todo se construye sobre los cimientos dejados por intentos anteriores. Veamos algunos de ellos.

Se considera que estas son las propuestas de valor fundamentales de SOA: reusabilidad, soporte para la automatización y monitoreo de procesos de negocios (en inglés, Business Process Automation o BPA y Business Process Monitoring o BPM) e interoperabilidad.

El reuso siempre ha sido una meta de la ingeniería del software, una meta hasta el momento no ha sido alcanzada. Se han propuesto diferentes técnicas, y si bien todas han aportado a la solución, ninguna resolvió el problema por completo. Algunas de ellas son: la programación orientada a objetos, los componentes de software, los patrones de diseño, el uso de frameworks y, finalmente, el reuso de servicios y procesos.

La automatización y el monitoreo de procesos también tienen sus antecedentes. Debemos mencionar algunos de los intentos realizados en la última década para estandarizar los lenguajes de definición de procesos (base fundamental para lograr la automatización):

- BPML Business Process Modeling Language (iniciativa de la BPMI -Business Process Management Initiative)

- XPDL XML Process Definition Language (iniciativa de la WfMC – Workflow Management Coalition)

- WSCI Web Services Choreography Interface (iniciativa de SUN)

- ebXML Extensible Binary Meta Language (Meta Lenguaje Binario Extensible) (iniciativa de OASIS/UN)

- WSFL (Web Service Flow Language - iniciativa de IBM)

- XLANG (iniciativa de Microsoft)

- BPEL4WS Business Process Execution Language for Web Services (más conocido como BPEL, surge de la unión del WSFL de IBM, y el XLANG de Microsoft)

Por último, algo muy parecido ocurre con la integración de aplicaciones y la interoperabilidad. Podemos encontrar varios enfoques tendientes a lograr la interacción entre aplicaciones implementadas en múltiples lenguajes y/o plataformas.

Algunos de ellos son: DCE (Data Communication Equip - Equipo de Comunicación de datos), CORBA (Common Object Request Broker Architecture - arquitectura común de intermediarios en peticiones a objetos), EDI (Electronic Data Interchange – intecambio electrónico de datos), DCOM (Distributed Component Object Model - Modelo de Objetos de Componentes Distribuidos) , EJB (Enterprise JavaBeans), EAI (Enterprise Application Integration) y Web Services.

3.3 ¿Por qué ahora va a funcionar?

La intención de este trabajo investigativo es dejar en claro que SOA es, ante todo, un concepto, un modelo arquitectónico que propone un cierto paradigma y un conjunto de lineamientos por seguir.

Este paradigma se puede ser implementar de diversas maneras. El uso de Web Services es una de ellas; la más popular, y probablemente, la más completa. Es por eso que es válido hablar de Web Services antes de responder la pregunta: ¿por qué ahora va a funcionar?

El surgimiento de los Web Services, y la gran cantidad de trabajo que se realiza en la generación y estandarización de protocolos para estos servicios tendientes a resolver temas tales como transaccionalidad, seguridad, etc. (lo que algunos han dado en llamar el Web Services Framework o WSF), son algunos de los motivos que hacen pensar en la implementación de arquitecturas orientadas a servicio utilizando Web Services como algo factible.

Este proceso de estandarización está haciendo que todo lo relacionado con SOA en general y con Web Services en particular adquiera la adhesión que no lograron otros intentos, como CORBA y DCE. Los estándares definidos se están

convirtiendo en estándares adoptados, y eso es lo que le da la fuerza real a cualquier iniciativa.

Por supuesto, éstos no son los únicos motivos. Considerando que la dinámica del mercado hasta hace algún tiempo, no requería de estandarización por parte de los grandes proveedores de plataformas, sino, más bien, todo lo contrario. De este modo, las soluciones de integración tradicionales (del tipo EAI) son, en general, propietarias. Esto implica un alto costo en licencias y, también, la contratación de personal técnico especializado, lo cual encarece los costos.

Una vez mas los costos vuelven a ser un imperativo. La necesidad de incrementar el alcance y el rango de las soluciones de integración actuales – con la reducción simultánea de los costos asociados a la implementación, es lo que impulsa a la industria hacia la estandarización, el uso de software libre (Open Source) y la evolución de las Arquitecturas Orientadas a Servicio

CAPITULO 4
ARQUITECTURA

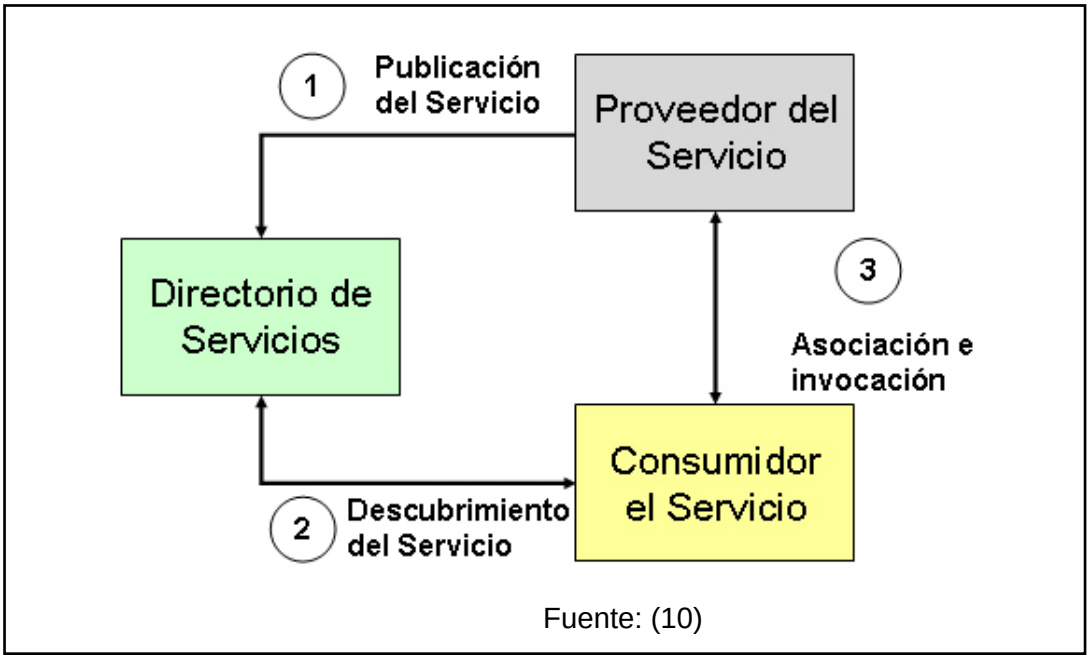
4.1 Arquitectura SOA

Se puede imaginar a SOA, de una manera sencilla, como una arquitectura consistente en proveedores y consumidores de servicios. (Figura 5)

Los proveedores definen el formato de los servicios y el modo de invocarlos a través de una interfaz independiente de la implementación. Los consumidores usan esta interfaz para construir los datos necesarios para invocar estos servicios, y realizan la invocación.

Además, y como un componente opcional, puede existir un directorio que actúa como intermediario, en el cual los proveedores publican sus interfaces y desde el cual los consumidores las "descubren". Este tipo de directorio de servicios puede ser de gran utilidad en organizaciones que tengan una gran cantidad de servicios.

Figura 5. Arquitectura Básica de SOA



Fuente: (10)

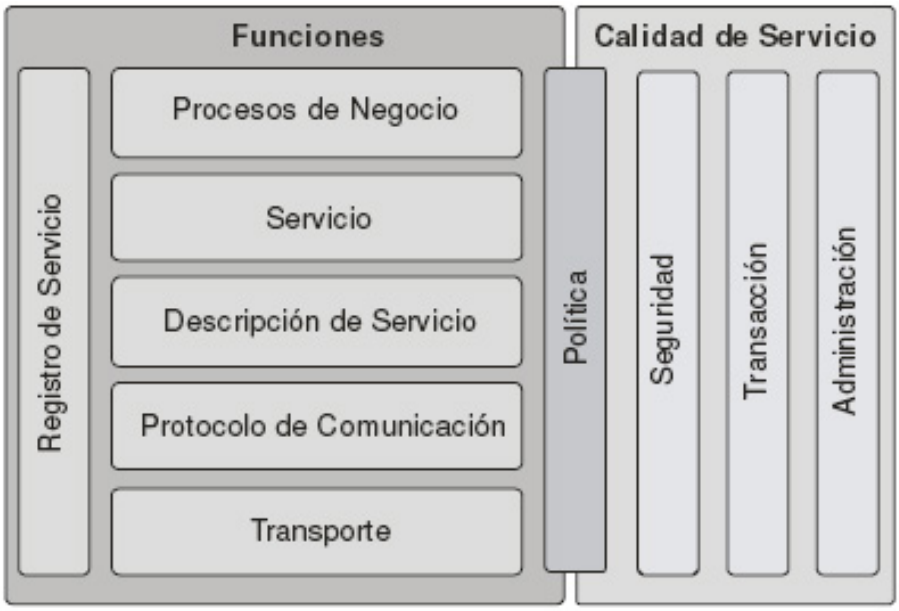
A partir de este modelo como base, podemos construir servicios que tengan distinto nivel de granularidad: desde servicios de infraestructura básicos (acceso a datos,

seguridad, etc.), hasta aplicaciones – expuestas como servicios –, pasando por servicios de negocio reutilizables.

La arquitectura SOA presenta una forma de construir sistemas distribuidos que entreguen a la aplicación funcionalidad, como servicios para aplicaciones de uso final u otros servicios.

En la (Figura 6) se muestra un esquema de los distintos niveles y los elementos que podrían observarse en una arquitectura orientada a servicios.

Figura 6. Elementos de una arquitectura orientada a servicios (SOA)



Fuente: (9)

Se muestran dos diferentes zonas dentro de los elementos de SOA, una que se refiere los aspectos funcionales de la arquitectura y otra que abarca aspectos de calidad de servicio. A continuación se describen brevemente:

Funciones

- Transporte: es el mecanismo utilizado para llevar las demandas de servicio desde un consumidor de servicio hacia un proveedor de servicio, y las respuestas desde el proveedor hacia el consumidor.

- Protocolo de comunicación de servicios: es un mecanismo acordado a través del cual un proveedor de servicios y un consumidor de servicios comunican qué está siendo solicitado y qué está siendo respondido.
- Descripción de servicio: es un esquema acordado para describir qué es el servicio, cómo debe invocarse, y qué datos requiere el servicio para invocarse con éxito.
- Servicios: describe un servicio actual que está disponible para utilizar.
- Procesos de Negocio: es una colección de servicios, invocados en una secuencia particular con un conjunto particular de reglas, para satisfacer un requerimiento de negocio.
- Registro de Servicios: es un repositorio de descripciones de servicios y datos que pueden utilizar proveedores de servicios para publicar sus servicios, así como consumidores de servicios para descubrir o hallar servicios disponibles.

Calidad de Servicio

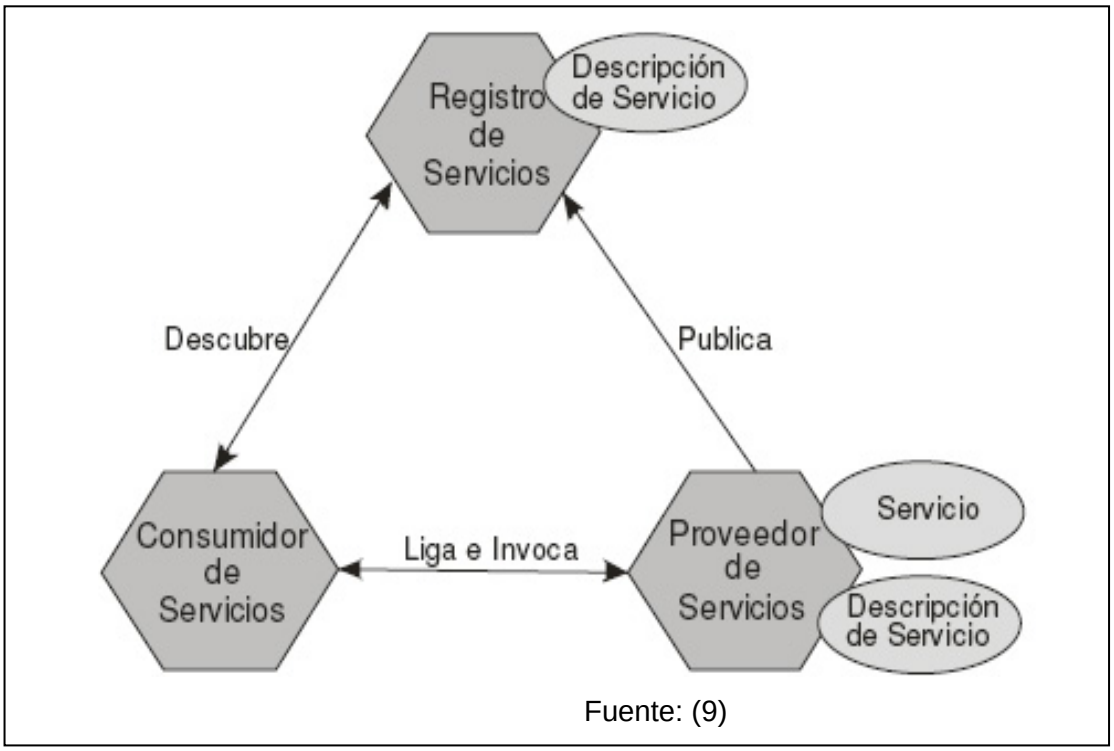
- Política: es un conjunto de condiciones o reglas bajo las cuales un proveedor de servicio hace el servicio disponible para consumidores.
- Seguridad: es un conjunto de reglas que pueden aplicarse para la identificación, autorización y control de acceso a consumidores de servicios.
- Transacciones: es el conjunto de atributos que podrían aplicarse a un grupo de servicios para entregar un resultado consistente.
- Administración: es el conjunto de atributos que podrían aplicarse para manejar los servicios proporcionados o consumidos.

4.1.1 Colaboraciones en SOA

Las colaboraciones en SOA siguen el paradigma find, bind and invoke, (descubrir, publicar e invocar) donde un consumidor de servicios realiza la localización dinámica de un servicio consultando el registro de servicios para hallar uno que cumpla con un determinado criterio. Si el servicio existe, el registro proporciona al consumidor la interfaz de contrato y la dirección del servicio proveedor.

La siguiente figura ilustra las entidades (roles, operaciones y artefactos) en una arquitectura orientada a servicios, donde estas colaboran.

Figura 7. Colaboraciones en SOA



Fuente: (9)

"Cada entidad puede tomar el rol de consumidor, proveedor y/o registro:

• Un consumidor de servicios es una aplicación, un módulo de software u otro servicio que requiere un servicio, y ejecuta el servicio de acuerdo a un contrato de interfaz.

• Un proveedor de servicios es una entidad direccionable a través de la red que acepta y ejecuta consultas de consumidores, y publica sus servicios y su contrato de interfaces en el registro de servicios para que el consumidor de servicios pueda descubrir y acceder al servicio.

• Un registro de servicios es el encargado de hacer posible el descubrimiento de servicios, conteniendo un repositorio de servicios disponibles y permitiendo visualizar las interfaces de los proveedores de servicios a los consumidores interesados." (9)

Las operaciones son:

- Publicar o Registrar. Para poder acceder a un servicio se debe publicar su descripción para que un consumidor pueda descubrirlo e invocarlo.
- Descubrir o Localizar. Un consumidor de servicios localiza un servicio que cumpla con un cierto criterio consultando el registro de servicios.

- Ligar e Invocar. Una vez obtenida la descripción de un servicio por parte de un consumidor, éste lo invoca de acuerdo a la información en la descripción del servicio.

Finalmente, los artefactos en una arquitectura orientada a servicios son:

- Servicio. Un servicio que está disponible para el uso a través de una interfaz publicada y que permite ser invocado por un consumidor de servicios.
- Descripción de servicio. Una descripción de servicio especifica la forma en que un consumidor de servicio interactuará con el proveedor de servicio, especificando el formato de consultas y respuestas desde el servicio. Esta descripción también puede especificar el conjunto de pre-condiciones, pos-condiciones y/o niveles de calidad de servicio (QoS - Qualitity of Service).

4.1.2 Capas de la arquitectura SOA

Las capas en las que se estructura la arquitectura orientada a servicios son: (1)

Capa 1: Sistemas operacionales. Esta capa contiene sistemas o aplicaciones existentes, incluyendo aplicaciones ERP (Planificación de Recursos Empresariales) sistemas heredados e implementaciones de sistemas orientados a objetos, así como aplicaciones de inteligencia de negocio. SOA puede reutilizar sistemas existentes e integrarlos utilizando técnicas de integración orientadas a servicios.

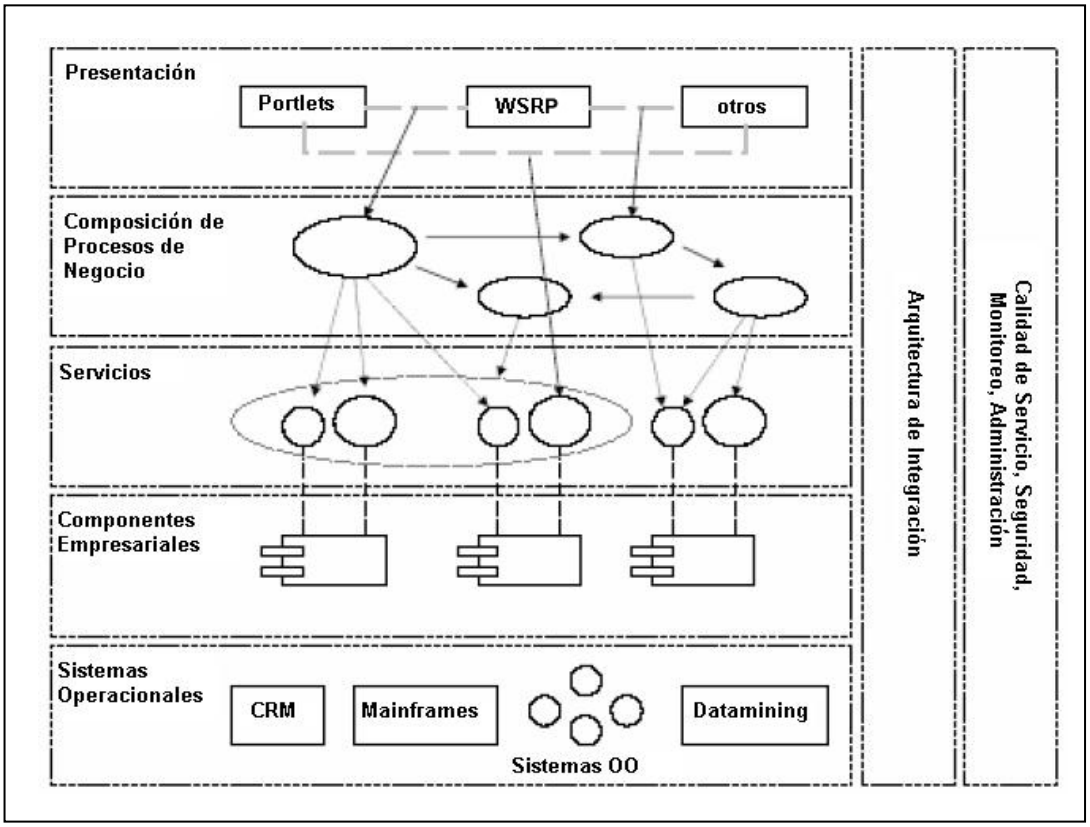
Capa 2: Componentes empresariales. Es la capa encargada de realizar la funcionalidad y mantenimiento de la calidad del servicio de los servicios expuestos.

Capa 3: Servicios. Los servicios de negocio residen en esta capa. Pueden ser descubiertos o pueden ser enlazados estáticamente y después invocados en servicios compuestos. Esta capa de exposición de servicios también permite obtener componentes empresariales (de la capa anterior), componentes de unidades de negocio, y en algunos casos componentes específicos del proyecto y exteriorizar un subconjunto de sus interfaces en forma de descripción de servicios. Los servicios existen aislados o en servicios compuestos.

Capa 4: Composición de procesos de negocio. Esta capa define la composición de procesos de negocio a partir de los servicios de la capa anterior. Los servicios se introducen en un flujo a través de orquestación y coreografía, para ejecutar un proceso de negocio. Se pueden utilizar herramientas visuales de composición de flujos para el diseño.

Capa 5: Presentación o acceso. Normalmente esta capa estaría fuera del ámbito de la SOA, pero debido a recientes estándares como Web Services for Remote Portlets versión 2.0 (OASIS Web Services for Remote Portlets 2003) se están convirtiendo en relevantes los servicios en la capa de presentación.

Figura 8. Capas y componentes de una SOA Fuente (1)



Capa 6: Integración de servicios (ESB) Posibilita la integración de servicios a través de la introducción de un conjunto fiable de capacidades como: enrutamiento inteligente y otros mecanismos de transformación, normalmente descritos como Enterprise Service Bus (ESB).

Capa 7: Calidad del servicio. Esta capa proporciona las capacidades necesarias para monitorizar, gestionar y mantener propiedades de calidad del servicio, como

seguridad, ejecución y disponibilidad. Es un proceso de background a través de mecanismos de sentir-responder y herramientas para monitorear la salud de las aplicaciones SOA incluyendo las implementaciones de WS-Management (Web Services for Management) y otros protocolos y estándares relevantes que implementan la calidad de servicio para una SOA.

4.2 Enterprise Service Bus (ESB)

"Se trata de una implementación de arquitectura SOA, altamente distribuida, dirigida por eventos, cuyo último objetivo es la integración de aplicaciones heterogéneas y distribuidas. Es una plataforma basada en estándares que combina mensajería, servicios web, transformación de datos y enrutamiento inteligente para conectar y coordinar la interacción de un número significativo de aplicaciones de empresas extendidas con integridad transaccional". (9).

La infraestructura de comunicaciones, conocida como Enterprise Service Bus (ESB), debe ser la que provee de servicios estándar, mediante los cuales se permite la publicación y la localización de servicios.
Adicionalmente, deberá ofrecer: (9)

Independencia de lenguaje-plataforma: Debe permitir la conexión de servicios implementados en cualquier tecnología. Al dar soporte a los métodos y mecanismos estándares para desarrollar e interconectar aplicaciones a través de la empresa, tales como WSDL, SOAP, los ESB reducen drásticamente el tiempo de implementación y el coste total de propiedad de los proyectos de integración.

Arquitecturas dirigidas por eventos: Se basa en utilizar eventos como disparadores que inician la distribución de un mensaje que informa a varios receptores acerca de un evento para que realicen las acciones pertinentes. Estas arquitecturas tienen las siguientes características:

- Desacopladas: la aplicación o servicio que envía el mensaje no sabe quienes están suscritos a ese tipo de mensaje. La relación se establece por la información, es decir por el mensaje, pero no entre las aplicaciones.
- Mensajería publicar/suscribir: ciertos sistemas publican información acerca de algún tipo de evento en la red para que otros sistemas (que se han suscrito y

por tanto autorizado a recibir este tipo de mensajes) puedan recibir dicha información y actuar en consecuencia.

- Asíncronas: Soporta interacciones asíncronas donde la información se envía sin que se espere una respuesta inmediata o se necesite mantener una conexión activa entre los dos sistemas mientras se espera la respuesta.

Un ESB debe permitir el soporte de diferentes tipos de mensajería.

Transformación de mensajes: Debe permitir la modificación de mensajes de modo que puedan ser adaptados al formato esperado por el receptor.

Direccionamiento inteligente: El bus debe ser capaz de determinar los destinatarios de un mensaje, basándose en su contenido. En un ESB los datos se pasan entre los sistemas conectados al bus utilizando mensajes. La coordinación de mensajes se realiza mediante un concepto denominado enrutamiento basado en itinerarios.
Un itinerario de un mensaje es un metadato que viaja junto al mensaje y proporciona la lista de direcciones siguientes a alcanzar por el mensaje.
El itinerario es un conjunto de instrucciones que le dice al framework de ejecución del ESB a qué sistemas se tiene que enviar el mensaje y este viaja de sistema a sistema a través del bus.

Orquestación de servicios: Debe otorgar facilidades para la coordinación de servicios. Los ESB incluyen la capacidad de configurar, implementar y administrar los servicios. A diferencia de un servidor de aplicaciones centralizado y monolítico o de las arquitecturas intermediarias (broker) de integración, los ESB proporcionan una óptima flexibilidad y, más aún, permiten una administración y escalabilidad de los servicios a nivel independiente, con el fin de alcanzar la mayor eficiencia operativa posible. La transparencia de ubicación permite que los servicios se actualicen, muevan o reemplacen sin necesidad de modificar los códigos de las aplicaciones

Establecimiento y monitoreo de niveles de servicio: Debe permitir el establecimiento de niveles de servicio (Service Level Agreement o SLA) y monitorear su cumplimiento.

Auditoria: Debe permitir el seguimiento de los mensajes e invocaciones realizadas a través del bus. Además proporcionar capacidades de auditoria a nivel tecnológico

del estado del bus y de los servicios disponibles en el bus. Esto se logra mediante la implantación de distintas políticas de autenticación y autorización

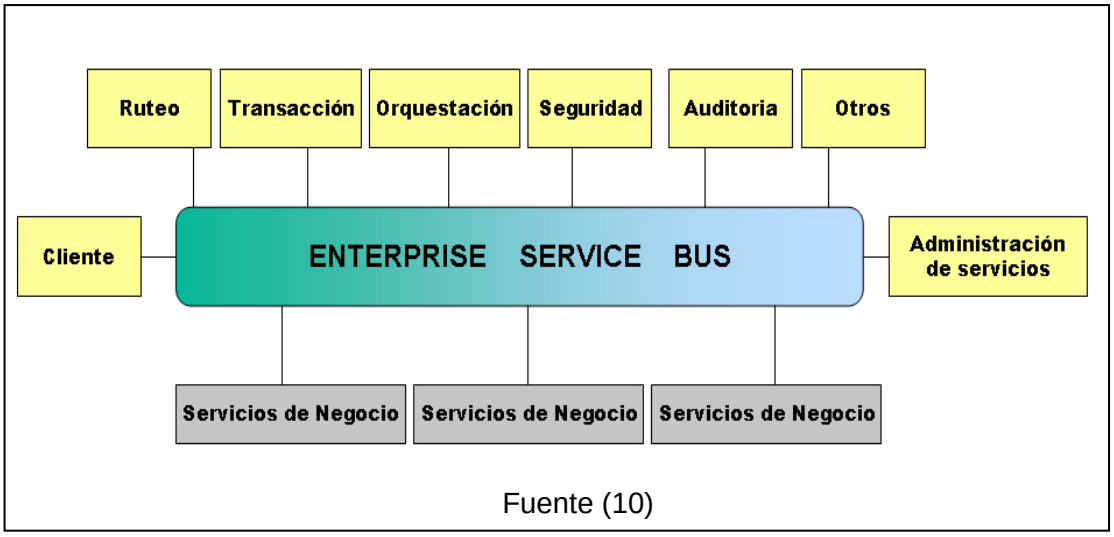
El ESB es más una construcción lógica que un producto en si. Como podemos observar en la lista anterior, es bastante difícil satisfacer todos estos requerimientos con un único producto, sobre todo, si tenemos en cuenta que las necesidades de integración no son sólo intra-organizacionales sino también inter-organizacionales.

Esto suele dificultar la adopción de un único producto más allá de los límites de la organización.

Esta construcción lógica se logrará a medida que los servicios de infraestructura, las aplicaciones corporativas y otros productos evolucionen hacia el soporte de un conjunto de estándares que les permitan interoperar: el framework de Web Services.

Como en el caso de SOA, el ESB no es un concepto nuevo. Cualquier herramienta, framework o lenguaje que soporte la funcionalidad básica de registrar, buscar y vincular servicios permite la construcción de un ESB simple. Los productos de tipo EAI (Enterprise Application Integration), CORBA, DCE y Java RMI son algunas alternativas de implementación.

Figura 9. Funciones e infraestructura de un ESB



Fuente (10)

Por lo tanto, todas ellas adolecen de uno o más de los problemas mencionados anteriormente: son soluciones propietarias, son dependientes de la plataforma, son

dependientes de un lenguaje o generan un alto nivel de acoplamiento. Un ESB que permita el grado de interoperabilidad requerido en el mercado actual debe basarse en estándares y no estar sujeto a ningún requerimiento de lenguaje o sistema operativo.

Este grado de estandarización e independencia puede lograrse con implementaciones de ESB construidas sobre las bases del framework de Web Services. Si a los Web Services se les adiciona el trabajo que se está llevando a cabo con la creación y estandarización de especificaciones y protocolos que permitan resolver cuestiones tales como seguridad, autenticación, orquestación y otras, el framework de Web Services se transforma en una plataforma natural para la implementación de un ESB estándar e interoperable.

CAPITULO 5
WEB SERVICES

5.1 Concepto y funcionamiento

“Un servicio Web (en inglés Web service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web”. (11)

Una definición de SW sería la emitida por la W3C: “una aplicación software identificada por una URI, cuyas interfaces y vinculaciones son capaces de ser definidas, descritas y descubiertas como artefactos XML. Un SW soporta la interacción con otros agentes software mediante el intercambio de mensajes basado en XML a través de protocolos basados en Internet”. (2)

5.1.1 Características de los Web Services

- Los Web services están basados en los estándares: XML y HTTP (el transporte más usual de WS) están ampliamente aceptados y son soportados por casi todos los proveedores de software.
- Además son independientes del lenguaje y de la plataforma: Los formatos de datos utilizados por WS están basados en XML, y son independientes del lenguaje y de la plataforma de implementación.
- Pueden atravesar los firewalls corporativos: Los protocolos de transporte usados por WS (HTTP y SMTP por ejemplo) generalmente son soportados por los firewalls corporativos.
- Son auto descriptivo: El Web Services Description Language (WSDL) brinda al proveedor de servicios un mecanismo que le permite describir a sus consumidores potenciales la manera en que éste debe ser invocado y el formato de los parámetros que espera.

Los web services comunican aplicaciones, permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado las aplicaciones, cuál sea el sistema operativo o la plataforma en que se ejecutan y cuáles los dispositivos utilizados para obtener acceso a ellas. La comunicación se caracteriza por el intercambio de mensajes XML y por ser independientes del protocolo de comunicación. Para lograr esta independencia, el mensaje XML se envuelve de manera apropiada para cada protocolo gracias a la creación del protocolo de transporte SOAP (Simple Object Access Protocol). (8)

El lenguaje WSDL (Web Services Description Languaje), expresado en XML, describe cómo acceder al servicio, de qué funciones dispone, qué argumentos necesita y qué devuelve cada uno.

La otra tecnología fundamental implicada es UDDI (Universal Description, Discovery, and Integration). UDDI es un directorio de servicios web donde se puedan publicar los servicios ofrecidos, dar características del tipo de servicio, y realizar búsquedas.

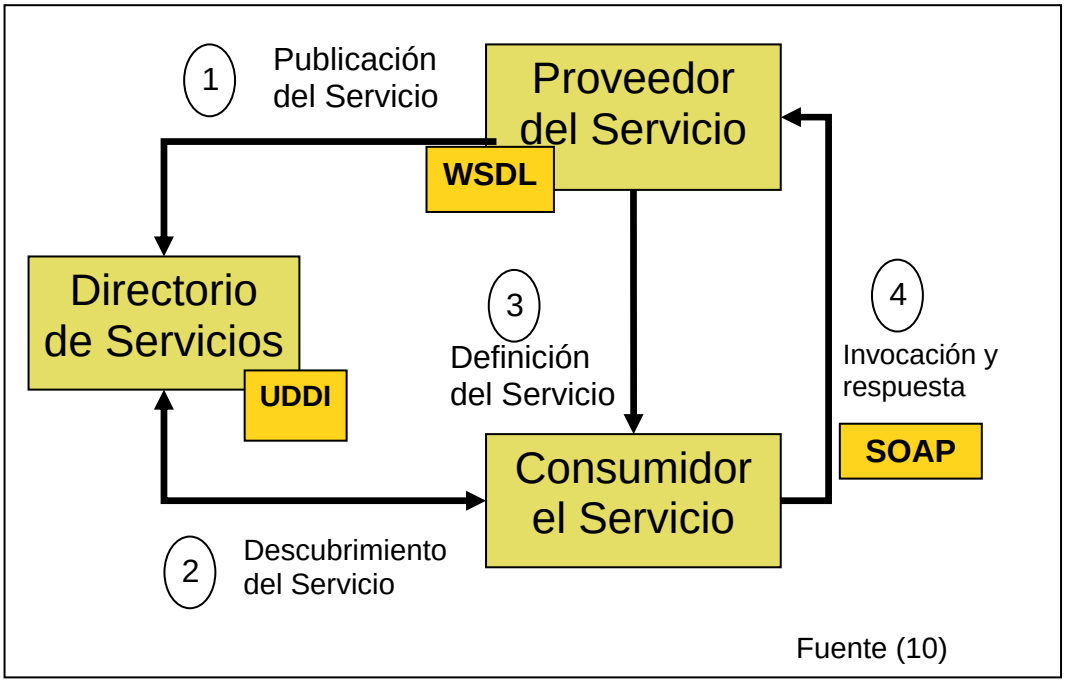
En definitiva, SOAP define un protocolo XML para la interoperabilidad básica entre servicios, WSDL introduce un lenguaje común para describir servicios y UDDI provee la infraestructura requerida para publicar y descubrir servicios. Figura (10).

5.1.2 Arquitectura de los Servicios Web

La arquitectura de servicios Web está basada en las interacciones entre tres elementos: proveedor de servicio, registro o directorio de servicios y consumidor del servicio. Las interacciones comprenden las operaciones de publicación, búsqueda y enlace.

En un escenario típico, un proveedor de servicios aloja un modulo de software accesible por la red (una implementación de servicio Web). El proveedor de servicios define una descripción de servicio para el servicio Web y lo publica en un registro de servicios. El consumidor del servicio utiliza una operación de búsqueda para encontrar la descripción del servicio localmente o desde el registro de servicio y utiliza la descripción del servicio para enlazarse con el proveedor del servicio e invocar o interactuar con la implementación del servicio Web.

Figura 10. Arquitectura de los servicios Web



En una arquitectura de servicios web se distinguen los siguientes roles:

Proveedor del servicio: Desde una perspectiva de negocio es el propietario del negocio. Desde una perspectiva arquitectural, es la plataforma que aloja el servicio.

Demandante del servicio: Desde una perspectiva de negocio es la aplicación que requiere satisfacer ciertas funciones. Desde una perspectiva arquitectural, es la aplicación que busca e invoca o inicia una interacción con el servicio. El papel del demandante del servicio puede ser jugado por un navegador dirigido por una persona o un programa sin interfaces de usuario por ejemplo otro servicio web.

Registro del servicio: Este es un registro donde buscar las descripciones de servicios, donde los proveedores de servicios publican sus descripciones de servicios. Los demandantes de servicios encuentran el servicio y obtienen información de enlace en la descripción del servicio durante el desarrollo para el enlace estático y durante ejecución para el enlace dinámico. Para servicios accedidos estáticamente, el registro del servicio es opcional en la arquitectura, porque un proveedor de servicio puede enviar la descripción directamente al demandante del servicio. Así mismo demandantes de servicios pueden obtener una descripción de otras fuentes.

Cualquier aplicación que utilice servicios web debe realizar las siguientes operaciones:

- Publicación de la descripción del servicio: Para que el demandante de servicios pueda encontrar el servicio, es necesario que la descripción del servicio sea publicada.
- Búsqueda de la descripción del servicio: En la operación de búsqueda el demandante de servicio puede recuperar directamente una descripción de servicio o puede consultar un registro de servicios por el tipo de servicios requerido. Esta operación puede llevarse a cabo en dos fases diferentes: en tiempo de diseño, para recuperar la descripción de la interfase del servicio para el desarrollo del programa y en tiempo de ejecución, para realizar el enlace e invocación del servicio.
- Enlace o invocación del servicio: En la operación de enlace el demandante del servicio invoca o inicia una operación con el servicio en tiempo de ejecución usando la descripción del servicio para localizar, contactar e invocar el servicio.

Estas operaciones pueden darse de forma independiente o de forma iterativa.

5.2 Protocolos y especificaciones

Los servicios web integran tres protocolos, cada uno orientado a resolver un problema particular:

- Web Services Description Language (WSDL): Describe el servicio.
- Universal Discovery Description Integration (UDDI): Se usa para describir la ubicación del servicio en la red. Es un formato mediante el cual se estandariza el “descubrimiento” de Web Services en la red. Existe una alternativa al uso de UDDI, ebXML, pero no es muy utilizada.
- Simple Object Access Protocol (SOAP): Es el protocolo de empaquetamiento de los datos de Web Services. Para transmitir los datos, suele usarse HTTP, pero podría emplearse cualquier protocolo.

Existe también un gran número de especificaciones conocidas como Web Server Extensions, que intentan estandarizar el manejo de aspectos tales como seguridad, mensajeria confiable, direccionamiento y enrutamiento, transaccionalidad, auditoria, etc. Estas especificaciones son manejadas por la organización WS-I (Web Services

Interoperability – Interoperabilidad de Servicios Web). A continuación indicamos un listado con algunas de ellas:

- WS-Security
- WS-SecureConversation
- WS-Policy
- WS-Trust
- WS-Choreography
- WS-Reliability
- WS-ReliableMessaging
- WS-DistributedManagement
- WS-Addressing
- WS-Attachments
- WS-Eventing
- WS-Transaction
- WS-Notification

5.2.1 WSDL

WSDL (Web Server description Language) es un documento XML que es utilizado para describir los mensajes SOAP, y como estos mensajes son intercambiados. Supongamos que hemos creado un Web Service y queremos que otras aplicaciones lo utilicen, las aplicaciones deberán acceder a un documento WSDL, en donde podrán conocer los métodos que expone nuestro Web Service y como acceder a ellos. (14)

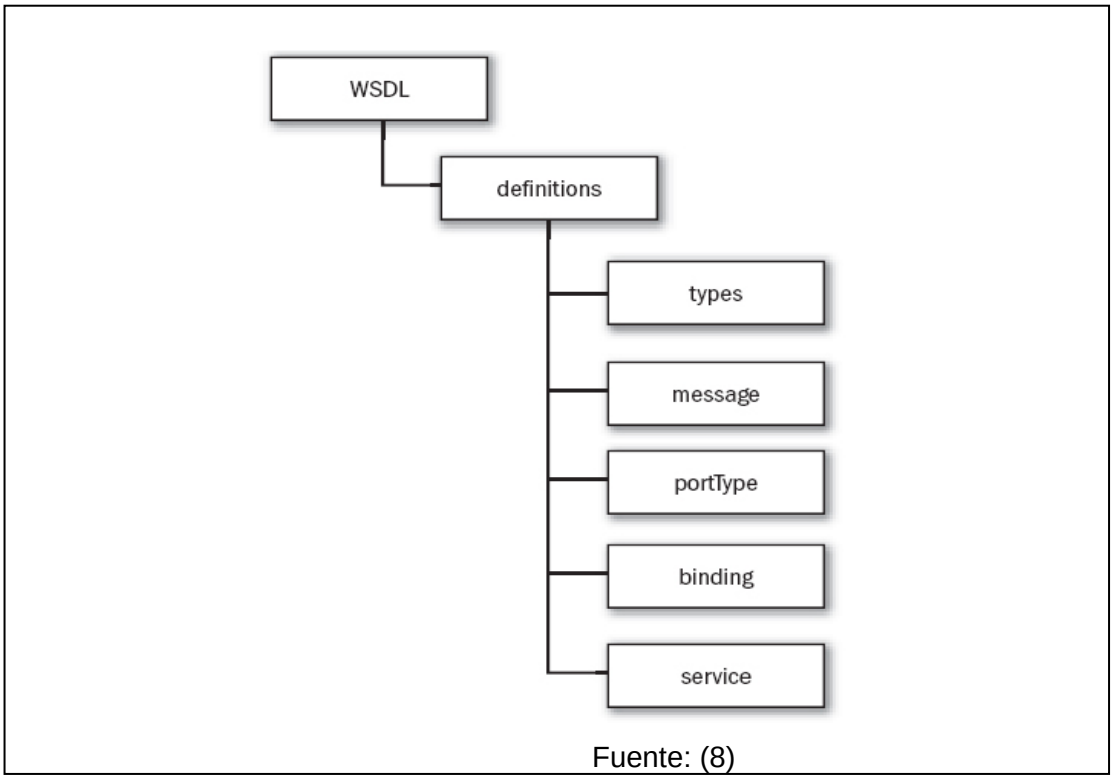
En esencia, WSDL es un contrato entre el proveedor del servicio y el cliente mediante el cual el proveedor del servicio indica:

• Qué funciones se pueden invocar
• Qué tipos de datos utilizan esas funciones
• Qué protocolo de transporte se utilizará para el envío y recepción de los mensajes (por lo general SOAP).
• Cómo acceder a los servicios. En esencia, mediante qué URL se utilizan los servicios.

5.2.1.1 Estructura de WSDL

Encontramos un elemento definitions en la raíz y diversas definiciones en su interior. Los servicios son definidos utilizando seis elementos principales:

Figura 11. Componentes WSDL



Fuente: (8)

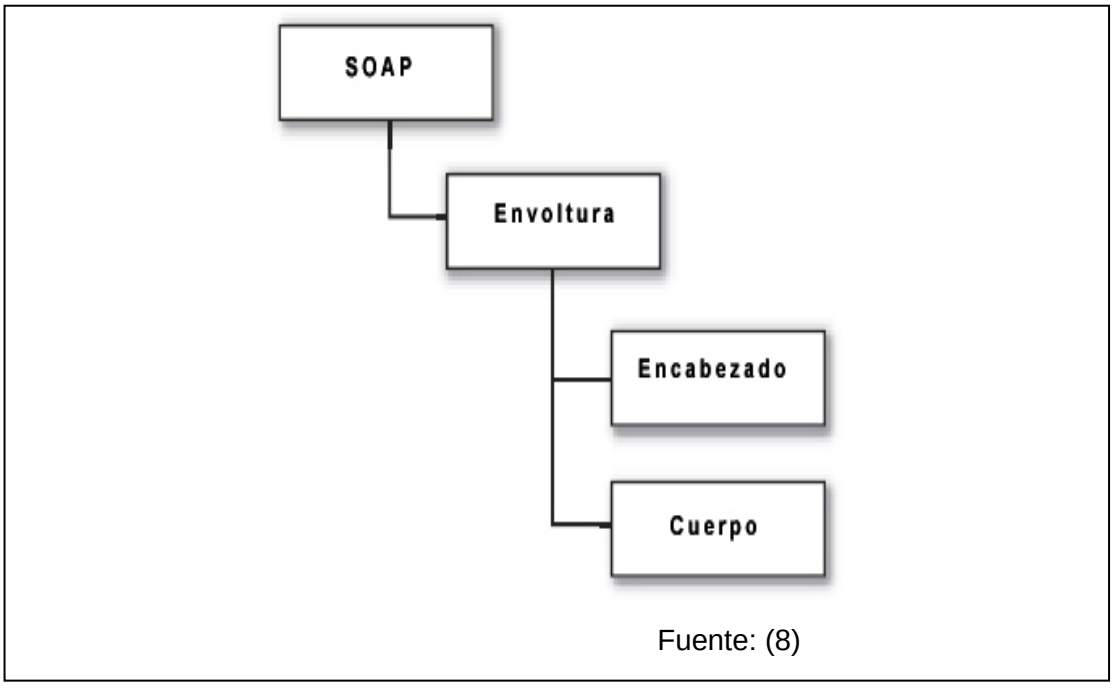
- types: proveen definiciones de los tipos de datos utilizados para describir los mensajes intercambiados.
- message: que representa una definición abstracta de los datos que están siendo transmitidos. Un mensaje de divide en una serie de partes lógicas, cada una de las cuales se asocia con alguna definición de algún sistema de tipos.
- portType: que es un conjunto de operaciones abstractas. Cada operación hace referencia a un mensaje de entrada y uno de salida.
- binding: que especifica un protocolo concreto y las especificaciones del formato de los datos de las operaciones y los mensajes definidos por un porType en concreto.
- port: que especifica una dirección para un binding, para así definir un único nodo de comunicación.
- service: que se utiliza para unir un conjunto de puertos relacionados.

5.2.2 SOAP

SOAP (Simple Object Access Protocol) es el protocolo base de los Web Services. Este protocolo esta basado en XML y no esta sujeto a ninguna plataforma o lenguaje de programación. SOAP es un protocolo que proporciona un mecanismo estándar de empaquetar mensajes. (12)

“Este protocolo está pensado para el intercambio de información en entornos descentralizados y distribuidos.
Usa las tecnologías relacionadas con XML a fin de definir un marco de trabajo extensible para los mensajes. Provee una estructura de mensajes capaz de ser intercambiada sobre una gran cantidad de protocolos de soporte. Este marco ha sido diseñado con el fin de que fuera independiente del cualquier modelo de programación y otras implementaciones de semánticas.” (8)
Una petición SOAP tiene tres partes: una envoltura, un encabezado (opcional) y un cuerpo, que puede ser un documento XML u otro tipo de datos.

Figura 12. Componentes SOAP



Fuente: (8)

Las ventajas que ofrece SOAP son las siguientes:

- No está asociado a ningún lenguaje.
- No se encuentra fuertemente asociado a ningún protocolo de transporte.
- No está ligado a ninguna infraestructura de objetos distribuidos.

- Aprovecha los estándares existentes en la industria como XML, el cual se toma para codificar los mensajes.
- Permite la interoperabilidad entre múltiples entornos.

5.2.3 UDDI

UDDI, “Universal Description, Discovery and Integration”, es un elemento central del grupo de estándares involucrados en la tecnología web services.

“UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros”. (12)

UDDI provee un método estandarizado para la publicación y el descubrimiento de información sobre los SW. Es el mediador a través del cual se conocen los posibles clientes con los proveedores de los servicios. Además se centra en el proceso de publicación y descubrimiento dentro de la arquitectura orientada a servicios.

Un registro UDDI ofrece un mecanismo basado en estándares para clasificar, catalogar y manejar servicios web de forma de que puedan ser descubiertos y consumidos por otras aplicaciones. Varios registros UDDI se pueden agrupar para formar un UBR (UDDI Business Registry) con la idea de que se apliquen las mismas políticas de autenticación, cifrado, o se balancee la carga de trabajo.
Los UDDI y los UBR pueden ser públicos o privados. Los privados permiten el registro de los servicios web sólo a sus miembros, añadiendo, por lo general, ciertas medidas de seguridad. Sin embargo, permiten la consulta de sus registros por cualquier usuario. La mayor parte de los UDDI y UBR existentes son privados.
Las empresas que implementan registros UDDI facilitan en general herramientas gráficas (vía Web o locales) para interactuar con el registro así como APIs (Application Programming Interface - Interfaz de Programación de Aplicaciones) para integrar con las aplicaciones.

Una organización puede registrar tres tipos de información dentro de un registro UDDI:
- Páginas blancas: Constituida por información básica de contacto e identificadores de la empresa, incluyendo el nombre del empresa, dirección, información de

contacto. Esta información permite a otros descubrir los SW de una empresa basándose en la identificación del negocio.
- Páginas amarillas: Consiste en información que describe al SW mediante diferentes categorizaciones o taxonomías.
- Páginas Verdes: Consiste en información técnica sobre los servicios que aportan las propias empresas, que describe el comportamiento y las funciones soportadas por un SW. Indican dónde están ubicados los SW.

5.2.4 XML

XML (eXtensible Markup Language - lenguaje de marcas extensible), es un metalenguaje (lenguaje para definir lenguajes) extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación de SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML (). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML (eXtensible Hypertext Markup Language), SVG (Scalable Vector Graphics), MathML (Mathematical Markup Language). (15)

Como objetivos de XML podemos mencionar los siguientes:

- Que fuera idéntico a la hora de servir, recibir, y procesar la información de HTML
- Que fuera normal y conciso.
- Que fuera extensible.
- Que fuese fácil de leer y editar.
- Que fuese fácil de implantar, programar y aplicar a los distintos sistemas.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un

papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Entre las ventajas de XML podemos mencionar las siguientes:

- Comunicación de datos. Si la información se transfiere en XML cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.
- Migración de datos. Si trabajamos en formato XML sería muy sencillo mover datos de una base de datos a otra.
- Aplicaciones Web. Con XML hay una sola aplicación que maneja los datos y para cada navegador podemos tener una hoja de estilo o similar para aplicarle el estilo adecuado.

5.3 Diferencias con otras alternativas

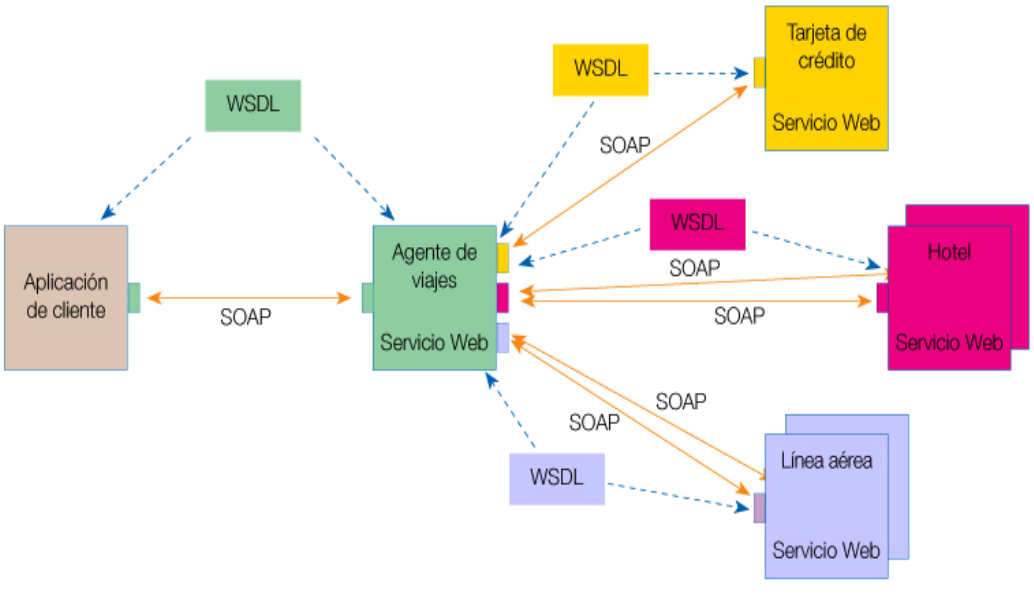
El termino Web Services no es un sinónimo de SOA y tenemos otras alternativas de implementación. Muchas de las diferencias de WS con respecto a las otras opciones se desprenden de lo escrito hasta ahora, aunque existen algunas más.

Resumiendo, las ventajas más importantes son la estandarización de los protocolos utilizados y la interoperabilidad natural que ofrecen algunos de ellos (fundamentalmente, XML y HTTP). Sin embargo, también hay desventajas. Es preciso tener en cuenta el esfuerzo extra que consume la serialización, la deserialización y el transporte de datos en formato XML. Como ya dijimos, algunas de las especificaciones tendientes a resolver problemas básicos aún están en etapa de desarrollo, con lo cual generan incertidumbre con respecto a si la tecnología podrá resolver estas cuestiones y cuál será el modo definitivo de hacerlo.

5.4 Ejemplo de Web Service

Según el ejemplo de la figura 13, un usuario (que juega el papel de cliente dentro de los Servicios Web), a través de una aplicación, solicita información sobre un viaje que desea realizar haciendo una petición a una agencia de viajes que ofrece sus servicios a través de Internet.

Figura 13. Ejemplo de Web Service



Fuente: (5)

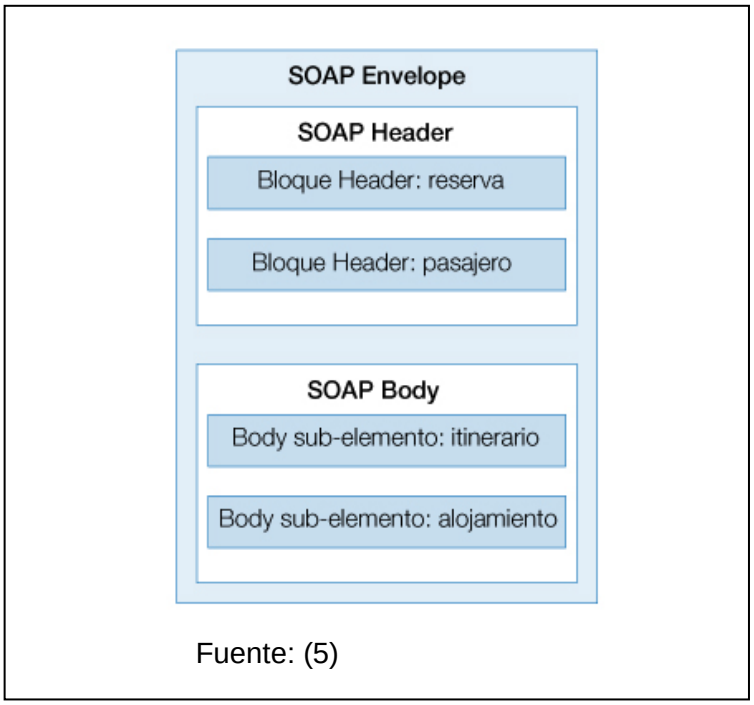
La agencia de viajes ofrecerá a su cliente (usuario) la información requerida. Para proporcionar al cliente la información que necesita, esta agencia de viajes solicita a su vez información a otros recursos (otros Servicios Web) en relación con el hotel y la línea aérea. La agencia de viajes obtendrá información de estos recursos, lo que la convierte a su vez en cliente de esos otros Servicios Web que le van a proporcionar la información solicitada sobre el hotel y la línea aérea.

Por último, el usuario realizará el pago del viaje a través de la agencia de viajes que servirá de intermediario entre el usuario y el servicio Web que gestionará el pago.

En todo este proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría SOAP (Protocolo Simple de Acceso a Objetos). Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP, etc. SOAP especifica el formato de los mensajes. El mensaje SOAP está compuesto por un envelope (sobre), cuya estructura está formada por los siguientes elementos: header (cabecera) y body (cuerpo).

Para el caso del ejemplo la estructura de los mensajes estaría conformada de la siguiente manera:

Figura 14. Estructura de mensajes



Fuente: (5)

Por otro lado, WSDL (Lenguaje de Descripción de Servicios Web), permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes.

A continuación se muestra el código que se utilizaría para solicitar información sobre un viaje:

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
  <m:reserva xmlns:m="http://empresaviajes.ejemplo.org/reserva"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
        env:mustUnderstand="true">
      <m:referencia>
```

```
            uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d
      </m:referencia>
    <m:fechaYHora>2001-11-29T13:20:00.000-05:00</m:fechaYHora>
   </m:reserva>
   <n:pasajero xmlns:n="http://miempresa.ejemplo.com/empleados"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
     <n:nombre>Pepe Ejemplo</n:nombre>
   </n:pasajero>
  </env:Header>
 <env:Body>
       <p:itinerario
          xmlns:p="http://empresaviajes.ejemplo.org/reserva/viaje">
        <p:ida>
          <p:salida>Nueva York</p:salida>
          <p:llegada>Los Angeles</p:llegada>
          <p:fechaSalida>2001-12-14</p:fechasalida>
          <p:horaSalida>última hora de la tarde</p:horaSalida>
          <p:preferenciaAsiento>pasillo</p:preferenciaAsiento>
        </p:ida>
        <p:vuelta>
          <p:salida>Los Angeles</p:salida>
          <p:llegada>Nueva York</p:llegada>
          <p:fechaSalida>2001-12-20</p:fechaSalida>
          <p:horaSalida>media-mañana</p:horaSalida>
          <p:preferenciaAsiento/>
        </p:vuelta>
      </p:itinerario>
      <q:alojamiento
        xmlns:q="http://empresaviajes.example.org/reserva/hoteles">
       <q:preferencia>ninguna</q:preferencia>
      </q:alojamiento>
    </env:Body>
  </env:Envelope>
```

CAPITULO 6
IMPLEMENTACIÓN DE SOA

6.1 Cuando adoptar SOA

Para llegar a una implementación de SOA, debemos analizar que tanto conviene adoptar una arquitectura de este tipo. Según podemos observar en muchas de las publicaciones y sitios de Internet dedicados a este tema, la respuesta es muy sencilla, siempre se debería adoptar SOA.

El modelo SOA, plantea un sistema distribuido, al menos en su implementación vía WS. Los sistemas distribuidos son, en general, mucho más complejos que los locales. Todo se hace más complicado, como ejemplo tendríamos la coordinación de transacciones. Entonces, lo primero que debemos pensar es si el problema que estamos intentando resolver necesita una arquitectura distribuida.

La arquitectura SOA intenta solucionar algunos problemas del negocio tales como una mayor necesidad de integración mediante herramientas y aplicaciones que se ejecutan en diversas plataformas/ lenguajes y además procesos de negocio cambiantes en los que la no adaptación de los sistemas de TI a estos cambios en tiempo y forma puede significar una gran pérdida económica para la organización. Considerando esto mientras nuestro negocio tenga estas necesidades, SOA será una alternativa a tomar muy en cuenta.

Nuestra empresa u organización debería ser modelada como una colección de servicios, esto no lleva a un cambio en la manera de concebir los sistemas, para así lograr obtener los mayores beneficios de SOA. Se debe planear una forma de llegar a esa meta, para lo cual lo primero es que nuestra organización quiera llegar hacia ella.

Si los niveles gerenciales no llegan a convencerse de los beneficios de SOA, y si además no se cuenta con el apoyo de arquitectos y desarrolladores, cualquier iniciativa de adopción de SOA no podrá ser llevada a feliz termino.

6.2 Camino hacia SOA

En el caso de que, luego de un análisis, lleguemos a la conclusión de que en nuestra organización, necesitamos una arquitectura de este tipo y que por lo tanto la misma está dispuesta a asumir este desafío, se plantea la siguiente interrogante ¿por dónde debemos empezar?

Debemos llevar a cabo un proceso incremental, con etapas bien definidas, probablemente seleccionando alguna aplicación que pueda obtener los máximos beneficios de este enfoque. Para lograr este objetivo debemos sortear muchos obstáculos que deben ser salvados en el camino.
La implementación de SOA es un proceso de larga duración que no se logra de un día para otro.

El camino hacia SOA tiene varias etapas, cada una de las cuales implica superar una cierta curva de aprendizaje y maduración. Estas etapas son: arquitectura de integración accidental, servicios encapsulando aplicaciones, servicios administrados y, finalmente, cambio de paradigma.

6.2.1 Arquitectura de integración accidental

En esta etapa podemos observar una arquitectura totalmente orientada a aplicaciones autocontenidas, en las que tenemos una integración punto a punto. Cada una implementa sus propios servicios de infraestructura (logging, seguridad, etc.).

En esta etapa se debe llevar a cabo una correcta planeación antes de empezar a crear servicios y conectarlos, para lo cual debemos plantearnos las siguientes inquietudes:

¿Qué es un servicio?
¿Qué servicios se requieren?
¿Qué servicios se deben desarrollar?
¿Cómo crear nuevos servicios?
¿Qué protocolo de comunicaciones usar para invocar los servicios?
¿Cómo administrar los servicios?
¿Cómo estandarizar los mensajes que van a intercambiar los servicios?

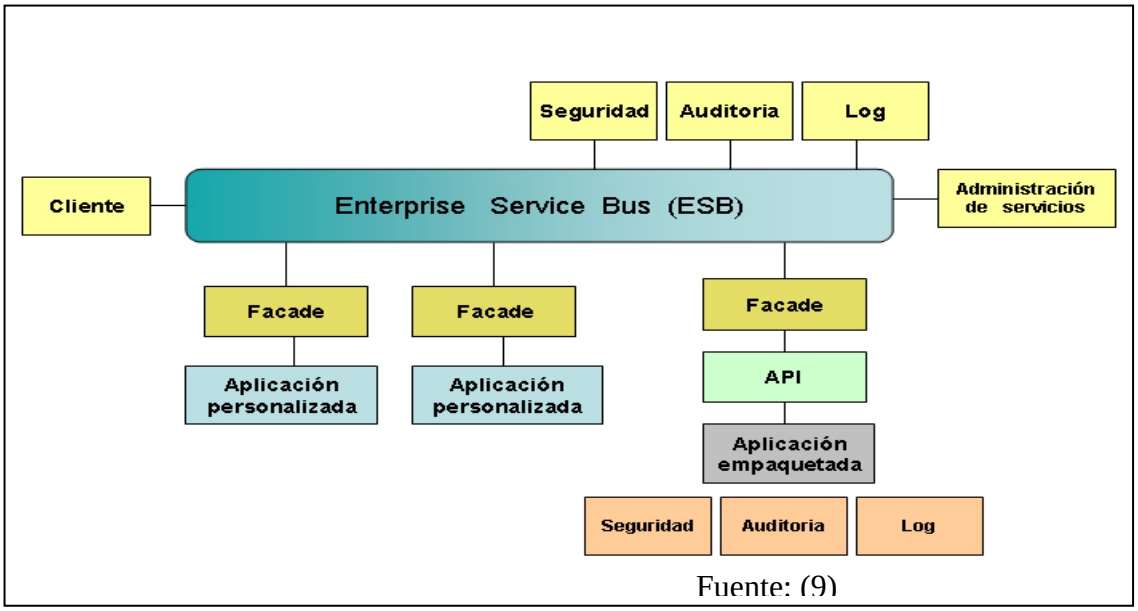
6.2.2 Servicios encapsulando aplicaciones

El primer paso en la adopción de SOA es implementar servicios que encapsulen las aplicaciones existentes, las cuales podrán ser accedidas a través de un protocolo común. Si bien es un estado relativamente simple de alcanzar (siempre y cuando contemos con un ESB que otorgue las capacidades mínimas de interconexión), carece de muchas de las características de una arquitectura de integración exitosa (transformación de datos, administración y monitoreo de servicios, y reuso de servicios).

6.2.3 Servicios Administrados

Se incorporan servicios de infraestructura básica en esta etapa tales como monitoreo, autenticación, registro, transformación de datos, etc. Estos servicios podrán ser accedidos desde cualquier aplicación cliente, de modo que se da la posibilidad de agregar administración a nuestra capa de servicios.
Además, algunas funcionalidades de infraestructura propias de las aplicaciones pueden ser refactorizadas como servicios: seguridad, auditoria, etc. Esta refactorización implica modificaciones al código de las aplicaciones. Muchas de ellas (especialmente, las “empaquetadas”) pueden no estar diseñadas para permitir el reemplazo de estos servicios básicos; por lo tanto, tendremos un esquema híbrido en el que algunas aplicaciones utilizan servicios de infraestructura comunes, mientras que otras no.

Figura 15. Refactorización de servicios



Fuente: (9)

Algunos servicios básicos pueden refactorizarse, siempre que las aplicaciones estén preparadas para hacerlo. En general, esto no será tan difícil de lograr con aplicaciones “empaquetadas”.

El estado final (y los máximos beneficios) se alcanza cuando todos los procesos de la organización son modelados como una colaboración de servicios.

Continuamos avanzando hacia SOA, pero estos avances se reflejan sólo en el área de TI. El negocio (sus procesos y reglas) sigue distribuido y replicado en diferentes aplicaciones.

6.2.4 Cambio de paradigma

Es el último escalón evolutivo de una SOA. Un verdadero enfoque orientado a servicio se logra cuando las aplicaciones monolíticas son descompuestas en servicios de negocios autocontenidos y bien definidos, esto es el desarrollo de aplicaciones basadas en procesos.

En esta etapa el aspecto clave es que toda la funcionalidad de infraestructura es provista mediante servicios. Los procesos de negocio son finalmente replicados como una composición de servicios, coordinada y orquestada por otro servicio.
Se llega a encapsular la complejidad de los procesos, compartiendo el estado entre clientes y manipulando procesos de larga vida.

Además se debe realizar un eficiente monitoreo de procesos. Al ejecutar procesos, es posible monitorearlos para analizarlos y poder encontrar posibilidades de mejora en los mismos

Podemos considerar esta etapa como la mas difícil de lograr en una implementación de SOA.

6.3 Dificultades

Las dificultades en el rediseño y de la reimplementación de nuestras aplicaciones son muchas, pero debemos tomar en cuenta otras: (9)

- Cambio de los estándares: Muchos de los estándares aún se encuentran en evolución, con lo cual pueden ocasionar incompatibilidades entre productos que

soportan distintas versiones, que nos pueden llevar a la modificación de código existente para adaptarlo a una nueva versión del estándar.

- Inmadurez de las herramientas de desarrollo: La evolución constante de los estándares implica que muchas herramientas tengan que suplir las falencias de éstos, lo que ocasionara problemas a medida que los que los estándares maduren.

- Normalización Sintáctica: Cuando se trabaja con decenas de sistemas, es normal que no todos manejen el mismo vocabulario. Para algunos, un mismo objeto, por ejemplo Cliente" se puede representar de manera distinta. Es importante que durante el modelado de los procesos se intente normalizar la estructura de los objetos de negocios, para simplificar la integración de sistemas y reducir la necesidad de realizar transformaciones en las que se pueden perder datos, al interconectar diversos sistemas.

- Confiabilidad de la red: En una arquitectura de servicios totalmente distribuida, cualquier falla en la red puede ocasionar una catástrofe. La infraestructura de redes debe ser redundante y robusta, y esto puede requerir importantes inversiones en tiempo y en dinero.

- Desempeño de los servicios Web: La serialización, la transmisión y la deserialización de datos en formato XML puede resultar costosa en tiempo y recursos. Cabe destacar que la serialización es el proceso de convertir un objeto de forma que pueda transportarse fácilmente. Por ejemplo, puede serializar un objeto y transportarlo por Internet utilizando HTTP entre un cliente y un servidor. En el otro extremo, la deserialización reconstruye el objeto a partir de la secuencia.

- Cultura centrada en las aplicaciones: Los procesos y las prácticas utilizados en el desarrollo deben ser revisados. Un cambio en el código no afecta a una sola aplicación sino a muchas. Esto requiere una percepción más global por parte de los grupos de desarrollo, junto con procesos que garanticen que los servicios desarrollados sean compatibles con las aplicaciones internas y externas a la organización que puedan consumirlos.

- Actualización tecnológica: Con mucha frecuencia conocemos de nuevos estándares, protocolos, herramientas y frameworks, sobre todo en el área de WS

y SOA. Por lo tanto tenemos que mantenernos actualizados, ya que esto nos permitirá decidir que cambios debemos implementar.

6.4 Recursos SOA

6.4.1 Herramientas para SOA

Muchas de las opciones de herramientas SOA que encontramos en el mercado son muy similares entre si, por lo tanto nos centraremos en el análisis de cuatro de estos productos que se consideran entre los más populares:

Oracle SOA

Proveedor: Oracle
www.oracle.com/technologies/ Suite soa/soa-suite.html
Características:

- Ambiente de desarrollo integrado (con capacidades de modelado, implementación y despliegue de soluciones)
- Motor de orquestación compatible con BPEL
- Administración de Web Services
- Motor de reglas de negocio
- Monitoreo y administración de actividades y procesos de negocio
- ESB basado en estándares que permite exponer sistemas existentes como servicios
- Directorio de servicios compatible con UDDI v3

WebSphere Business Integration Solutions

Proveedor IBM
www.ibm.com/software/websphere
Características:

- Ambiente de desarrollo integrado (con capacidades de modelado, implementación y despliegue de soluciones)
- Monitoreo y administración de procesos de negocio
- ESB basado en JMS

WebLogic

Proveedor BEA
www.bea.com/framework.jsp?CNT=products
htm&FP=/content/solutions/soa/solutions

- Plataforma de creación de servicios declarativa y programática
- Orquestación, monitoreo y administración de servicios
- ESB con soporte para varios tipos de mensajería
- Servicios de infraestructura estándar

Mule

Proveedor: SimphonySoft
http://mule.codehaus.org

- Licencia gratuita, aun para uso comercial
- ESB compatible con J2EE 1.4
- Orquestación de servicios compatible con BPEL
- Soporte programático y declarativo de transacciones
- Direccionamiento de mensajes programático, declarativo, basado en reglas y en el contenido

Aparte de las características de cada una de las herramientas mencionadas anteriormente, gran parte de las posibilidades que encontramos en el mercado incluyen siguientes características:

- Proveen un service bus basado en el framework de Web Services.
- Brindan servicios de ruteo basado en contenido y transformación de mensajes.
- Soportan los protocolos de transporte más comunes (HTTP, FTP, SMTP, JMS, etc.).
- Incluyen algún tipo de API que permite extenderlos, si es necesario.
- Permiten algún nivel de orquestación de servicios.

Es importante tomar en cuenta los siguientes puntos antes de seleccionar tal o cual producto, son mencionados a continuación:

- Visión y orientación del producto: ¿Es un producto genérico o está orientado a algún nicho particular del mercado? ¿Soporta todo tipo de interacciones o está centrado en algún conjunto de ellas?
- Costo: ¿Es comercial u Open Source? ¿Requiere la contratación de consultores especializados? Si es Open Source, ¿existen licencias de soporte?
- Madurez: ¿En qué etapa de desarrollo se encuentra? ¿Qué tipo de soporte brinda? ¿Existe una comunidad de usuarios activa?
- Completitud: ¿Tiene todas las características que necesitamos?
- Estandarización: ¿Son soluciones propietarias o respetan estándares?
- Productividad de desarrollo: ¿Provee un ambiente de desarrollo? ¿Se integra con los ambientes y frameworks de desarrollo más populares?
- Portabilidad: ¿Puede ser ejecutado sobre distintos servidores de aplicación, bases de datos, etc.?
- Interoperabilidad: ¿Genera implementaciones con algún tipo de dependencia hacia alguna plataforma/lenguaje?
- Monitoreo y administración de procesos: ¿Permite monitorear procesos en ejecución? ¿Permite la creación de tableros de control a medida? ¿Qué facilidades de administración de procesos en ejecución ofrece?
- Servicios de infraestructura: ¿Provee autenticación, autorización, transaccionalidad, auditoria, notificaciones, logging, etc.?
- Directorio de servicios: ¿Ofrece un repositorio donde publicar y ubicar dinámicamente los servicios?

Se debe notar que esta lista puede y debe ser completada en base al sentido común al momento de evaluar una herramienta, teniendo en cuenta las necesidades particulares del negocio.

CAPITULO 7
CONCLUSIONES Y RECOMENDACIONES

.

La tecnología de Web Services parece ser la única alternativa de implementación de SOA, que permite los niveles de estandarización e interoperabilidad necesarios actualmente. Sin embargo, se trata de una tecnología que aún está en desarrollo, con muchas especificaciones en etapa de desarrollo y que tiene algunos problemas.

Empresas líderes del mercado, como Microsoft, Sun e IBM, se encuentran desarrollando especificaciones que pueden ser usadas para construir bloques de SW más poderosos, seguros, confiables y transaccionales. Estas especificaciones ayudarán a las empresas a desarrollar aplicaciones usando SOA, de forma más sencilla.

Una vez publicados los estándares, se prevé que aumenten las herramientas de desarrollo para aplicaciones basadas en Web Services.

Debemos estar concientes que en la actualidad, ningún avance tecnológico tiene un impacto más profundo en las tecnologías de la información y los procesos de negocio que el surgimiento de SOA.

SOA permite que las organizaciones combinen fluidamente sus activos de TI existentes para desarrollar nuevas aplicaciones, procesos y modelos de negocios internos o externos a la empresa.

El objetivo de SOA es aportar para lograr el desarrollo de servicios que puedan ser reutilizados por la organización. Estas capacidades permiten que las tecnologías respondan rápidamente a los cambios de los negocios, los escenarios competitivos, las alianzas y las necesidades de los consumidores. SOA permite la creación de aplicaciones compuestas que trabajan a través de una única interfaz, facilitando la comprensión de distintos procesos de negocios

A lo largo de este trabajo investigativo hemos abordado el tema de Arquitectura Orientada a Servicios (SOA), tratando en lo posible de definir SOA, el porque de utilizar SOA, sus antecedentes, su arquitectura, su implementación. Por lo tanto es muy importante como recomendación tomar en cuenta los siguientes puntos adicionales para lograr este objetivo:

Determinar cuándo una organización está lista para SOA

Los problemas que pueden llevarnos a una implementación SOA generalmente son vistos como necesidades del negocio, ya que en una primera fase se debe analizar desde el punto de vista del negocio antes que lo tecnológico.

Cuando una organización decide hacer uso de una arquitectura SOA es porque tiene objetivos específicos de negocio que cubrir, reducir costos, aumentar ingresos, mejorar la productividad, comunicación inter-empresarial con varias empresas y ajustar los sistemas a los requerimientos del negocio.

Es recomendable un proceso de formación que vaya de arriba hacia abajo, es decir, partiendo de una educación a los directivos sobre las ventajas fundamentales de SOA y los beneficios que supone desplegarla.

Tener claro qué no es SOA

- Un producto o un proyecto aislado.
- Una única tecnología de software.
- Características de hardware.
- Una nueva idea.
- La solución para resolver todos sus problemas de IT.

Lo que se considera un proyecto SOA ideal

- Reconoce el valor de la integración de personas, procesos e información, con las prioridades del negocio.
- Trata de aportar valor para el negocio desde los activos ya presentes de IT.
- Reaccionar de forma rápida a los cambios de mercado.
- Información en tiempo real para la toma de decisiones.
- Incluye una estructura para organizar y administrar servicios a través de la organización.
- Incluye un repositorio de servicios para catalogar y almacenar los servicios.

- Favorece la reutilización
- Hace uso de las arquitecturas de tecnología existentes, para proveer los elementos fundamentales requeridos para construir una Arquitectura Orientada a Servicios robusta

Como conclusión final podemos acotar que SOA brinda el marco conceptual sobre el que se pueden gestar nuevos modelos tecnológicos, que respondan eficientemente a las crecientes necesidades tecnológicas del mundo empresarial actual. Sobre este modelo están evolucionando nuevos modelos en tecnologías de información y sistemas empresariales. Además que a pesar de que la operación de un sistema diseñado bajo la visión de SOA es compleja, las ventajas que aporta esta nueva arquitectura superan ampliamente los inconvenientes de la misma.

GLOSARIO

API	Application Programming Interface - Interfaz de Programación de Aplicaciones. Es un conjunto de especificaciones de comunicación entre componentes software.
BPA	Business Process Automation - Automatización de procesos de negocios.
BPEL4WS	Business Process Execution Language for Web Services - Lenguaje de Ejecución de Procesos de negocio para Servicios Web. Es un lenguaje basado en XML que describe formalmente los procesos de negocio y protocolos de interacción. BPEL4WS define como se conectan entre sí los servicios Web a fin de realizar una determinada tarea.
BPM	Business Process Monitoring - Monitoreo de procesos de negocios.
BPML	Business Process Modeling Language (iniciativa de la BPMI - Business Process Management Initiative). Similar a BPEL4WS en la construcción de flujos que definen el comportamiento de un Servicio Web. Utiliza WSCI para definir los intercambios entre los Servicios Web que forman parte del proceso.
CORBA	Common object request broker architecture - (arquitectura común de intermediarios en peticiones a objetos). Es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.
COREOGRAFIA	Permite trazar las secuencias de mensajes que se suceden entre todas las partes participantes del proceso de negocio en lugar de centrarse en los mensajes que intercambian los SW que implementan los procesos de negocio ejecutados únicamente por una parte.
DCE	Data Communication Equip - Equipo de Comunicación de datos. Equipos a través de los cuales conectamos los diferentes equipos de computación, a las líneas de comunicación.
DCOM	Distributed Component Object Model - Modelo de Objetos de Componentes Distribuidos. Es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí.

E A I	Enterprise Application Integration - Integración de Aplicaciones Empresariales. EAI es la integración de nuevas aplicaciones con las ya existentes, incluyendo las aplicaciones heredadas o los paquetes de software, de forma que todas juntas proporcionen las funcionalidades necesarias para soportar los procesos de negocio de la empresa. Esta integración permite a la organización mantener el ritmo de los cambios del mercado y reaccionar a tiempo frente a ellos.
ebxML	Extensible Binary Meta Language (Meta Lenguaje Binario Extensible) (iniciativa de OASIS/UN).
EDI	Electronic Data Interchange – intecambio electrónico de datos. Es un software Middleware que permite la conexión a distintos sistemas empresariales como ERP o CRM. El Intercambio Electrónico de Datos puede realizarse en distintos formatos.
EJB	Enterprise Java Beans. Es una arquitectura que define la forma de construir componentes distribuidos del lado del servidor. Esta tecnología garantiza que los componentes programados sean escalables, eficaces y seguros.
Firewalls	Un cortafuegos (o firewall en inglés), es un elemento de hardware o software utilizado en una red de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas según las políticas de red que haya definido la organización responsable de la red.
Frameworks	Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
IT	Information Technology - Tecnología de la información.
Java RMI	Es un mecanismo ofrecido en Java para invocar un método remotamente.
Math ML	Mathematical Markup Language. Es un lenguaje de marcado basado en XML, cuyo objetivo es expresar notación matemática de forma que distintas máquinas puedan entenderla, para su uso en combinación con XHTML en páginas web, y para intercambio de información entre programas de tipo matemático en general.
Metadado	Literalmente "sobre datos", son datos que describen otros datos. En general, un grupo de metadatos se refiere a un grupo de datos, llamado recurso.

Middleware	Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.
OMG	Object Management Group. Es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización NO lucrativa que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para tecnologías orientadas a objetos.
Open Source	Código abierto (del inglés open source) es el término por el que se conoce el software distribuido y desarrollado libremente.
ORB	Object Request broker. En Computación distribuida es el nombre que recibe una capa de software (también llamada middleware) que permite a los objetos realizar llamadas a métodos situados en máquinas remotas, a través de una red.
ORQUESTACION	Permite diseñar procesos de negocio ejecutables que puede interactuar (a nivel de mensaje) tanto con SW internos como externos. Con la orquestación, siempre representamos el control del proceso de negocio desde el punto de vista de una de las partes participantes que intervienen en el mismo.
Q o S	Qualitity of Service - Calidad de Servicio. Garantiza que se transmitirá cierta cantidad de datos en un tiempo dado.
SLA	Service Level Agreement - Acuerdo de Nivel de Servicio. Es una referencia a la hora de establecer parámetros de calidad del servicio.
SVG	Scalable Vector Graphics. Es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados.
UBR	UDDI Business Registry. El registro de empresas UDDI (UBR) es un directorio público global de empresas y servicios. Los usuarios plantean consultas en UBR para descubrir servicios web para obtener información sobre ellos.
UDDI	Universal Description, Discovery, and Integration. UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

WSCI	Web Services Choreography Interface (iniciativa de SUN). WSCI no define un flujo de tareas ejecutable. En su lugar, define un proceso mediante los intercambios de mensajes entre Servicios Web que participan en la realización del mismo.
WSDL	Web Services Description Languaje. Un formato XML que se utiliza para describir servicios Web.
WSF	Web Services Framework
WSFL	Web Service Flow Language - iniciativa de IBM
WS-I	Web Services Interoperability – Interoperabilidad de Servicios Web
XHTML	eXtensible Hypertext Markup Language
XLANG	Iniciativa de Microsoft
XPDL	XML Process Definition Language (iniciativa de la WfMC – Workflow Management Coalition)

BIBLIOGRAFIA

1. ARSANJANI. Ali. BORGES. Bernhard y Kerrie HOLLEY. "Service-oriented architecture". Web Service Journal. (2004). Vol. 4. No. 9. pp. 34-38.
2. Alonso, G., Casati, F., Kuno, H. y Machiraju, V., Web Services. Concepts, Architectures and Applications, Carey, M.J. y Ceri, S.(eds.). 2004, Berlin: Springer. Num. páginas: 354
3. Estado del Arte. Proyecto Batuta. Pablo Alvez, Patricia Foti, Marco Scalone http://www.willydev.net/descargas/WillyDev_ToolOrquesta.pdf
4. Fundamentos de la Arquitectura Orientada a Servicios. Javier Betancourt http://www.progresssoftware.com.mx
5. Guía Breve de Servicios Web http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb
6. Nuevas Tendencias en Sistemas de Información: Procesos y Servicios http://www3.unileon.es/pecvnia/pecvnia02/02_129_158.pdf
7. Principios de Web Services. WEBSERVICES www.tectimes.com/lbr/Graphs/revistas/lpcu104/capitulogratis.pdf
8. Servicios Web. Universidad de Castilla-La Mancha, España. http://www.info-ab.uclm.es/descargas/thecnicalreports/DIAB-05-01-1/Servicios%20Web.pdf
9. SOA: Un Modelo de Dominios que Ataca Todos los Aspecto de una Organización. Octavio Duré http://www.willydev.net/descargas/SOA/WillyDev_Nota_Revista_Code.pdf
10. Universo SOA. Paola Saavedra Martínez. http://www.syndeo.com.ar/Evento2006/Presentacion_PS_GeneXus_SOA.ppt
11. Wikipedia Servicios Web. http://es.wikipedia.org/wiki/Servicio_Web
12. Wikipedia SOAP http://es.wikipedia.org/wiki/SOAP
13. Wikipedia UDDI. http://es.wikipedia.org/wiki/UDDI
14. Wikipedia WSDL http://es.wikipedia.org/wiki/WSDL
15. Wikipedia XML http://es.wikipedia.org/wiki/XML
16. http://www..microsoft..com/usa/webcasts/ondemand/892.asp
17. Microsoft - Hacia una arquitectura empresarial basada en servicios http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art143.asp

DISEÑO DE MONOGRAFIA

1 TITULO DEL PROYECTO

“TECNOLOGIA SOA (ARQUITECTURA ORIENTADA A SERVICIOS)”.

2 SELECCIÓN Y DELIMITACIÓN DEL TEMA

Contenido: El presente trabajo de investigación se realizará dentro del área de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario. SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación. Se pretende evaluar esta tecnología tomando en cuenta las fortalezas y debilidades de la misma.

Espacio: El presente proyecto lo realizaré en la Universidad del Azuay en la ciudad de Cuenca.

Tiempo: La realización de esta monografía se llevara a cabo en un plazo no mayor a 7 semanas.

3 DESCRIPCIÓN DEL OBJETIVO DE ESTUDIO

El objetivo de este estudio en primer lugar es comprender la tecnología SOA para tener la capacidad de identificar los modelos de negocios a ser soportados y las soluciones coherentes a ser desarrolladas para soportarlas. Además cómo los procesos de negocios podrían beneficiarse a partir de esta tecnología.

Se pretende contar con material de referencia para consultas con respecto a este tema, para de esta manera poder llevar a la implementación de esta tecnología en las empresas interesadas.

4 RESUMEN DEL PROYECTO

La competencia intensa en un mercado global que cambia constantemente hace que el crecimiento de los ingresos a través de la introducción de nuevos productos, servicios y mercados sea una prioridad. Las empresas deben incrementar la

agilidad para responder rápidamente a las cambiantes condiciones de negocio. Los desafíos de negocio resultantes de este ambiente conducen a un conjunto de imperativos de IT que requieren de un entorno operativo basado en una arquitectura orientada a servicios (SOA).

SOA es la Arquitectura de Aplicaciones dentro de la cual cada función está definida como un servicio independiente con interfaces invocables muy bien definidas, que pueden conformar procesos de negocio al organizarse adecuadamente

5 INTRODUCCIÓN

En sus inicios, La Organización OMG (Object Management Group), generó la posibilidad de trabajar con programas que eran independientes de sus plataformas, proveedores, etc. convirtiéndose en los primeros Middleware, llamados en ese entonces ORB´s (Object Request broker) los cuáles se basaron en especificaciones CORBA (Common object request broker architecture).

Lamentablemente no todos los sistemas se desarrollaron usando esta filosofía y se incrementaron los problemas de integración.

La aparición de Java contribuyó a plataformas de programación neutral y XML contribuyó en datos que se auto describen en plataformas neutrales.

La aparición de Web Services eliminó algunas barreras, permitiendo la interconexión de aplicaciones mediante modelos orientado a objetos.

Esto permite que a cada servicio requerido se obtenga una respuesta:

- Sin importar donde están las aplicaciones.
- En que lenguaje fueron desarrolladas.
- La ruta que el mensaje debe seguir para ser exitoso

SOA es una Arquitectura o Plataforma para el diseño y desarrollo de sistemas que proveen servicios a otras aplicaciones a través de interfaces identificables y publicables, generando servicios que pueden ser invocados desde la red.

La implementación de una plataforma SOA que usa Tecnología Web Services genera nuevas formas de construir aplicaciones con modelos flexibles.

SOA es la propuesta de solución para dar respuesta a las necesidades de los negocios de adaptar soluciones tecnológicas flexibles y poder asumir los nuevos retos.

Es la base sobre la cual se pueden tomar los activos actuales y llevarlos a un nivel más alto y con ellos construir nuevos sistemas o aplicaciones.

6 SITUACIÓN ACTUAL Y FUTURA

SITUACION ACTUAL

- Aumento complejidad.
- Arquitecturas tradicionales que llegaron al límite de su capacidad de dar respuestas.

Por lo tanto se plantean las siguientes necesidades:

- Integración dentro de la empresa.
- Integración con aliados de negocio y clientes (B2B, B2C).
- Integración de plataformas fácil, rápida y segura.
- Aumentar el ROI.
- Mejorar tiempos de respuesta a necesidades del Negocio.
- Plataforma que permita el desarrollo de soluciones de forma rápida, integrables y re-usables.
- Plataforma que permita usar componentes y servicios existentes de forma dinámica y rápida, generación de soluciones.
- Competencia continua.
- Retener clientes (Fidelidad).

SITUACION FUTURA

Con esta investigación se pretende que sirva de material de consulta, además de dar a conocer la implementación, funciones, beneficios, y los aspectos mas importante de esta tecnología, logrando con esto dar una visión mas general de lo importante y lo útil que podría llegar a ser esta nueva arquitectura, la misma que poco a poco va ganando terreno en las organizaciones.

SOA brinda el marco conceptual sobre el que se pueden gestar nuevos modelos tecnológicos que respondan a las crecientes necesidades tecnológicas del mundo empresarial actual. Sobre este modelo están evolucionando los nuevos modelos en tecnologías de información y sistemas empresariales.

7 JUSTIFICACIÓN E IMPACTO

JUSTIFICACION

SOA no es un concepto nuevo, ya que desde hace más de 30 años la industria viene planteándose la necesidad de crear arquitectura de software débilmente acopladas. CORBA es muy probablemente la más cercana encarnación de este concepto.

Las Arquitecturas Orientadas a Servicios están motivadas por la creciente necesidad de los negocios de responder con rapidez a los cambios en el entorno comercial en que se desenvuelven. Esto los lleva a tener que cambiar sus sistemas tecnológicos con esa misma rapidez y para lograrlo es necesario que los componentes de esta infraestructura sean tan reutilizables y poco interdependientes que permitan una rápida reestructuración de los mismos.

Por ello, el motivo principal constituye el acercarnos al conocimiento y posterior manejo e implementación de esta arquitectura que se desarrolla vertiginosamente y será de gran aplicación en los próximos años a nivel mundial.

IMPACTO TECNOLOGICO

Es difícil hacer una recomendación general de arquitectura, sin embargo, podría decirse que SOA es una tendencia de arquitectura que ha llegado para quedarse. Por primera vez en mucho tiempo, existe un consenso en la industria sobre la importancia de un modelo de arquitectura particular y un apoyo tan generalizado para llevar las tecnologías hacia este modelo. Por tanto, podría decirse que más aún que decidir si pasarse o no a SOA, el hecho es que SOA llegará a las organizaciones tarde o temprano.

El impacto de SOA trascenderá en las organizaciones puesto que impulsa el alto desempeño ya que aprovecha las inversiones existentes, ofrece plazos de comercialización más rápidos, mitiga riesgos, proporciona una mejora continua, brinda transparencia en la ubicación y tiene independencia del protocolo y del dispositivo.

8 OBJETIVOS

Objetivo General

Proveer en base a esta investigación, información sobre SOA que propone un entorno compartido de trabajo, “crear servicios que sean reutilizables, y al mismo tiempo que estén unidos en un solo punto en la organización.

Objetivos Secundarios

Presentar al lector los siguientes beneficios de adoptar SOA

- Consolidar con esta tecnología un solo marco de referencia, nuevos desarrollos, plataformas existentes o nuevas adquisiciones de distintos proveedores, de forma consistente.
- Reducir los tiempos de diseño y desarrollo, entregando soluciones en menor tiempo.
- Al estar el negocio modelado en componentes, cualquier cambio en el diseño implica solamente reconfigurar los mismos, permitiendo mantener el control del proceso en todo momento.

Objetivos Personales

- Las Arquitecturas Orientadas al Servicio son frecuentemente mencionadas en las publicaciones de Tecnologías Informáticas, por ello lo considero un tema de gran importancia que amerita profundizar mis conocimientos al respecto.

9 MARCO TEÓRICO

Los elementos básicos que conforman SOA son:

- Proveedores de Servicios
- Consumidores de Servicios
- Bus Empresarial de Servicios

En realidad todo componente dentro de una organización puede ser tanto proveedor como consumidor de servicios. Todos los servicios interactúan entre si a través de un Bus Empresarial de Servicios (o ESB por sus siglas en ingles).

Además podemos definir otros elementos participantes dentro de una arquitectura SOA tales como:

•Cliente–proveedor

Se entiende Cliente como el componente que invoca un servicio provisto por un proveedor.

•Concepto de servicio

Un servicio es una unidad de trabajo realizada por un componente de software a fin de conseguir un resultado específico. El servicio debe ser alcanzable por parte de los consumidores a través de una interfaz programática.

• Utilización de webservices

La arquitectura orientada a servicios no especifica necesariamente que los servicios deben ser brindados a través de un protocolo específico. Los Web Services son en realidad un conjunto de estándares que definen un protocolo de

invocación remota de servicios, basados en HTML y XML. Si bien, son también un mecanismo adecuado y en muchos casos recomendable para implementar servicios no son el único. Es importante que las arquitecturas orientadas a servicios soporten múltiples protocolos a fin de cumplir al máximo su visión de brindar un modelo de integración para toda la plataforma tecnológica.

• Protocolos utilizados en SOA

-Web Services: que en realidad hacen referencia a un conjunto de protocolos, WSDL (Web Services Definition Language), XML, XMLP y SOAP (Simple Object Access Protocol), WOSA (Windows Open System Architecture). Existen además un conjunto amplio de protocolos que se sientan sobre estos para brindar características ampliadas de seguridad, confiabilidad y administración a los Web Services.

-UDDI: (Universal Description Discovery and Integration) que define el protocolo para describir, encontrar e integrar servicios.

-Adicionalmente, tal y como se mencionó anteriormente otros protocolos pueden ser utilizados como mecanismos válidos de invocación de servicios, por ejemplo, JMS (Java Messaging Service), CORBA, MQSeries, entre otros.

• Mensajes en un ambiente SOA

El mensaje es el contenido de la invocación del servicio que lleva la información propia del negocio necesaria para la realización de las operaciones atadas al servicio.

Otros elementos que forman parte de la arquitectura

El Enterprise Services Bus o ESB (Bus Empresarial de Servicios) es muy probablemente el componente más importante aún no mencionado. El Enterprise Services Bus se basa en la mejor práctica en patrones de diseño para integración de aplicaciones, típicamente conocido como HUB & SPOKE.

Este patrón plantea la existencia de un componente de mediación que provee servicios de ruteo, transformación de mensajes, publicación y distribución de eventos y soporte para múltiples protocolos.

10 CONTENIDOS

1.- Que es SOA?

- 1.1. Definición de Servicio
- 1.2. Propiedades de un Servicio.
- 1.3. Distintas cosas para distinta gente.

2.- Por que SOA?

- 2.1. Motivaciones y necesidades del negocio
- 2.2. Ventajas

3.- Mas allá del auge de SOA

- 3.1. Antecedentes
- 3.2. Diferencias con otras alternativas
- 3.3. Por que ahora va ha funcionar

4.- Arquitectura

- 4.1. Arquitectura SOA
- 4.2. Enterprise Service Bus (ESB)

5.- Web Services

- 5.1. Concepto, funcionamiento
- 5.2. Protocolo y especificaciones
- 5.3. Diferencias con otras alternativas

6.- Implementación de SOA

- 6.1. Cuando adoptar SOA
- 6.2. Camino hacia SOA
  - 6.2.1. La arquitectura de integración accidental
  - 6.2.2. Servicios encapsulando aplicaciones
  - 6.2.3. Servicios Administrados
  - 6.2.4. Cambio de paradigma
- 6.3. Dificultades
- 6.4. Recursos SOA
  - 6.4.1. Herramientas para SOA

7.- Conclusiones y Recomendaciones

8.- Anexos

11. PROCEDIMIENTOS METODOLÓGICOS

- Investigación de Arquitecturas Orientadas a Servicios
- Desarrollo del material de consulta.

12. RECURSOS HUMANOS Y TÉCNICOS

Recursos Humanos:

Investigador y Desarrollador: Carlos Xavier González Mejia

Asesores del Proyecto: Ing. Sist. Pablo Pintado

Recursos Materiales:

- Hardware: El hardware a utilizar es el siguiente:
- Portátil Acer Centrino 1.7 Ghz

Software

- Sistema Operativo Windows XP
- Acrobat reader
- Internet Explorer

13 CRONOGRAMA DE ACTIVIDADES

Para la elaboración de la monografía expuesta, se espera cumplir con el siguiente cronograma.

El tiempo viene dado en semanas.

"TECNOLOGIA SOA (ARQUITECTURA ORIENTADA A SERVICIOS)".

ACTIVIDADES	1	2	3	4	5	6	7
Recopilación de información previa							
Capítulos 1 y 2							
Capítulos 3 y 4							
Capítulos 5 y 6							
Conclusiones y revisión general							
Preparación de la presentación							
Entrega							

14 BIBLIOGRAFÍA

La bibliografía en la que apoyare la investigación del tema propuesto será.

http://www.huibert-aalbers.com/
20/11/2006

http://www.wilytech.com/
20/11/2006

"http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios"
25/11/2006

www.morfeo-project.org/index
25/11/2006

www.relecturas.com.ar/informes/soa/index
25/11/2006

www.celeritechsolutions.com/modules.php
25/11/2006

www.progresssoftware.com.mx/
26/11/2006

www.help400.es/
26/11/2006